National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

University of Alberta

Expert Systems Approach to Chemical Process Design and Operation

by

Randy Dong

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science,

in

Process Control

Department of Chemical Engineering

Edmonton, Alberta
Fall 1991

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-69925-6

Canada

Dr. Ming Rao
Dept. of Chemical Engineering
University of Alberta
Edmonton, AB
T6G 2G6

October 4, 1991

To whom it may concern:

I hereby give my release for any co-authored information that is contained within Randy Dong's thesis entitled:

Expert Systems Approach to Process Design and Operation.

There are two sections of his thesis which are based on three papers co-authored with myself. These papers are the following:

1) R. Dong and M. Rao. (1990) "A Learning Expert System for Phase Equilibrium Method Selection." Proc. of the 1990 Summer Computer Simulation Conference, Jul 6-8, 1990, Calgary, AB, Canada.

2) R. Dong, M. Rao, H. Bacon and V. Mahalec. (1991) "An Expert System for Refinery Plant Operation." Proc. of the IFAC Symposium on Distributed Intelligence Systems, Aug 13-15, 1991, Arlington, VA, USA.

3) R. Dong, M. Rao, H. Bacon and V. Mahalec. (1991) "An Expert Advisory System for Safe Plant Startup." Fourth International Symposium on Process Systems Engineering, Aug 5-9, 1991, Montebello, Que., Canada.

If there are any questions please contact me at (403) 492 - 8250.

Rao

Ming Rao
Professor of Chemical Engineering

University of Alberta

Release Form

Name of Author:	Randy Dong
Title of Thesis:	Expert Systems Approach to Chemical Process Operation and Design
Degree:	Master of Science
Year this Degree Granted:	1991

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

R. D.

Signed

Permanent Address:

847 Hooke Rd.
Edmonton, AB
T5A 4K5

Oct 4/91

Date

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled,

Expert Systems Approach to Chemical Process Design and Operation
submitted by
Randy Dong

in partial fulfillment of the requirements for the degree of Master of Science in Process Control.

Date: Oct 4/91

Dr. M. Rao, Supervisor

Dr. D. G. Fisher

Dr. H. Henein

Dr. V. Mahalec

Abstract

The objective of this research was to investigate the application of expert system technology to chemical process design and operation. This thesis includes two case studies. The first study illustrates the technique of machine learning in an expert system. The expert system provides recommendations for selection of vapor-liquid equilibrium (VLE) calculation methods. VLE calculations are used in the design and simulation of process equipment. This study extends the research performed by others in this area (Venkatasubramanian, 1985; Banares-Alcantara, Westerberg and Rychener, 1985; Gani and O'Connell, 1989). The expert system developed here used machine learning (Shavlik and Dietterich, 1990) to acquire knowledge. Example VLE data points were presented to the system. The system used several common calculation methods to predict the data points. The errors in prediction, along with a classification system based on chemical families, formed the criteria for VLE method selection.

Evaluation of the expert system was performed using about 400 experimental data points for training and testing. The expert system showed accuracies of 30% and 50% for two different selection objectives. No further gains in accuracy were shown after the first 100 training points. The degree of accuracy after 100 training points was approximately equal for both the training data set and the test set. Results also indicate that the VLE calculation methods are more competitive in predicting vapor phase compositions as compared to bubble-point temperatures. The architecture of the system was constructed so it could be used with any physical property which has several available prediction techniques.

The second expert system was developed to provide advice for the startup of a section of a refinery. It was developed in conjunction with an industrial partner. This expert system contained the startup knowledge from an industrial process operator with many decades of experience. The project was aimed at investigating the use of expert systems in process

operation as well as capturing this expert's knowledge prior to his retirement.

The expert system accepted input describing the current state of the unit and made recommendations on which startup procedures to perform. The selection of startup procedures was based on the current state of the unit and the knowledge captured from the expert operator.

Testing of the expert system within the plant indicated that the system could provide operation advice at the level of an expert. It made consistently correct recommendations. One element in the success of the system was the automation of the written manual. Hypertext (Gessner, 1990) was used to make access to the manual rapid and efficient. The hypertext system was integrated with the knowledge base to provide intelligent selection of appropriate manual sections. Another important result from this research was the development of a new conceptual process model. The startup problem needed a model for the chemical and mechanical states of the unit. The model is characterized by using the mechanical state, modified by the operators, to manipulate the chemical state. The chemical state is driven towards the goal state: the normal operation of the unit.

Acknowledgements

I would like to thank all those who helped me during my thesis research.

First of all, I would like to thank Dr. Rao, my advisor, and all members of the Intelligence Engineering Laboratory for all their advice and assistance.

The Department of Chemical Engineering was a key element in the first class training I received.

I would also like to thank ESSO Petroleum Canada Ltd., Strathcona Refinery, for providing partial financial support and an industrial environment for my thesis research.

Last, but not least, I would like to thank all members of my committee for taking the time to advise me on my thesis:

Dr. Crickmore, Chairman
Dr. Fisher, Committee Member
Dr. Henein, Committee Member
Dr. Mahalec, Committee Member
Dr. Rao, Committee Member and Supervisor.

Table of Contents

Chapter — Page

Abstract
Acknowledgements
Table of Contents
List of Tables
List of Figures
List of Symbols and Nomenclature
List of Abbreviations

1. Introduction 1
    1.1 Objectives 1
    1.2 Organization 3

2. Background on Expert Systems 4
    2.1 Definitions 4
        2.1.1 Artificial Intelligence 4
        2.1.2 Expert Systems 4
        2.1.3 Knowledge Representation 7
        2.1.4 Inference Engine 8
    2.2 Machine Learning 10

3. Case Study One: A Learning Expert System for Vapor-Liquid Equilibrium Calculations 12
    3.1 Introduction 12
    3.2 Background 13
        3.2.1 Previous Research in Expert Systems for Thermodynamics 13
        3.2.2 Vapor-Liquid Equilibrium 15
    3.3 System Description 22
        3.3.1 Chemical Classification 22
        3.3.2 Machine Learning 25
        3.3.3 Knowledge Representation 27
        3.3.4 Inferencing 31
        3.3.5 System Structure 33
        3.3.6 Implementation Platform 37
    3.4 Evaluation 37
        3.4.1 Illustrative Examples 43
        3.4.2 Evaluation of the Expert System 53
    3.5 Discussion 61

Chapter Page

3.6 Conclusions 65

4. Case Study Two: An Expert Advisory System for Safe Plant Startup 67
4.1 Introduction 67
4.2 Background 68
4.2.1 Process Description 69
4.2.2 Startup Procedures 71

4.3 System Structure 73
4.3.1 Chemical/Mechanical State 73
4.3.2 Knowledge Acquisition 74
4.3.3 Knowledge Representation 75
4.3.4 Hypertext 79
4.3.5 Program Structure 81
4.4 Evaluation of the Advisory System 85
4.4.1 Examples 87
4.4.2 Evaluation of the Expert System 89
4.5 Discussion 90
4.6 Conclusions 91

5. Conclusions 92

References 95

Appendices 99
Appendix A: List of the Pure Components Included in the VLE Expert System 99
Appendix B: List of the Production Rules for the VLE Expert System ... 103
Pure Component Classification Rules 103
VLEC Method Selection Rules 108
Appendix C: Source Code for the VLE Expert System 113
Appendix D: List of the Production Rules for the Plant Startup Expert System 164

List of Tables

Table		Page
3.1	Three Cubic Equations of State	17
3.2	Activity Coefficient Equations	21
3.3	Experimental Data	26
3.4	Pure Component Properties	35
3.5	Listing of the 15 VLEC Methods in this System	36
3.6	Summary of Calculation Results for Example Data Point 1	44
3.7	Summary of Calculation Results for Example Data Point 2	51
3.8	Average Number of 100% Correct Recommendations During 8 Test Sessions	58
4.1	Summary of the Startup Procedures	72
4.2	List of the Circuits in the Plant Unit	76

List of Figures

Figure	Page
2.1 Structure of an Expert System	6
3.1 Bubble-Point Calculation Algorithm	23
3.2 Classification Families for Chemical Compounds	24
3.3 Geometric Growth of the Production Rules	30
3.4 Complex Information Base	34
3.5 Classification Levels	38
3.6 Level 3 Error Matrix for Peng-Robinson Equation of State	39
3.7 Level 2 Error Matrix for Peng-Robinson Equation of State	40
3.8 Level 1 Error Matrix for Peng-Robinson Equation of State	41
3.9 C++ Classes Built for the Expert System	42
3.10 Effect of Example Data Point 1 on the Error Matrices	45
3.11 Inference Process for Methanol-Water Consultation after Training with Example Data Point 1	47
3.12 Inference Process for Butane-Pentane Consultation after Training with Example Data Point 1	48
3.13 Inference Process for Methanol-Benzene Consultation after Training with Example Data Point 1	49
3.14 New State of the Error Matrices after Training with Example Data Point 2	52
3.15 Inference Process for Methanol-Water Consultation after Training with Example Data Point 2	54
3.16 Inference Process for Butane-Pentane Consultation after Training with Example Data Point 2	55
3.17 Inference Process for Methanol-Benzene Consultation after Training with Example Data Point 2	56
3.18 Average Number of 100% Correct Recommendations During 8 Test Sessions	59
3.19 General Architecture of this Expert System	62
4.1 Simplified Schematic of the Process	70
4.2 Illustration of the Knowledge Structure	78
4.3 Illustration of the Rule Filters	80
4.4 Main Menu System	82
4.5 Startup Menu and Hypertext System	84
4.6 Phase and Procedure Hypertext System	84
4.7 Knowledge Processing Overview	86

List of Symbols and Nomenclature

a	Cubic Equation of State Parameter
a_i	Cubic Equation of State Parameter for component *i*
a_m	Cubic Equation of State Parameter for a mixture
A	Cubic Equation of State Parameter
A_{12}, A_{21}	Wilson Equation Interaction Parameters
b	Cubic Equation of State Parameter
b_i	Cubic Equation of State Parameter for component *i*
b_m	Cubic Equation of State Parameter for a mixture
B	Cubic Equation of State Parameter
f	Molar Fugacity
f_L	Molar Fugacity for Liquid Phase
f_V	Molar Fugacity for Vapor Phase
k_{ij}	Interaction Parameter for Cubic Equation of State
K	Equilibrium Ratio
K_i	Equilibrium Ratio for component *i*
P	Pressure
P_C	Critical Pressure
P_{VP}	Vapor Pressure
R	Gas Law Constant
T	Temperature
T_C	Critical Temperature
T_r	Reduced Temperature, T/T_C
u	Cubic Equation of State Parameter
v_{iL}	Molar Liquid Volume for component *i*
v_{jL}	Molar Liquid Volume for component *j*
V	Molar Volume
w	Cubic Equation of State Parameter
x	Liquid Phase Composition
x_i	Liquid Phase Composition for component *i*
y	Vapor Phase Composition
y_i	Vapor Phase Composition for component *i*
Z	Compressibility Factor
ϕ	Fugacity Coefficient
ϕ_L	Fugacity Coefficient for Liquid Phase
ϕ_V	Fugacity Coefficient for Vapor Phase
γ_i	Activity Coefficient for component *i*
$\Lambda_{12}, \Lambda_{21}$	Wilson Equation Interaction Parameters
ω	Acentric Factor

List of Abbreviations

AF&TT	Air Free & Tightness Test
AI	Artificial Intelligence
CAD	Computer Aided Design/Drafting
CF	Certainty Factor or Confidence Factor
CIB	Complex Information Base
DB	Database
DOS	Disk Operating System
EF	Evaluation Function
KB	Knowledge Base
MSW	Microsoft Windows(TM)
P&ID	Process & Instrumentation Diagram
VLE	Vapor-Liquid Equilibrium
VLEC	Vapor-Liquid Equilibrium Calculation

1. Introduction

This chapter introduces the objectives of this research as well as the organization of the thesis.

1.1 Objectives

Chemical engineering is a complex field where knowledge is gained through an initial stage of formal training and further by experience. Experienced engineers can rapidly troubleshoot engineering problems or supply designs which meet required specifications. Such experience can not be easily learned. If captured within an expert system, this experience could be made available for training junior engineers and for consultation in real engineering problems.

There are mainly two problems with implementing expert systems in chemical engineering. The first is a general problem encountered in any expert system development: how to capture the domain expertise and store it in a computer? Knowledge acquisition and organization is the major bottleneck in expert system development. The most common approach is that of the knowledge engineer. The knowledge engineer is a computer specialist who interviews the expert(s) to obtain the knowledge, and then encodes it within the computer. The knowledge engineer is usually ignorant of the domain knowledge (Luger and Stubblefield, 1989). Machine learning is a technique of allowing the computer to acquire knowledge automatically (Shavlik and Dietterich, 1990). Expert system shells are commercial computer programs supplying all parts of an expert system except the knowledge base. Shells can be used by domain experts to build and test a knowledge base. Another approach is to have the domain expert(s) directly build the expert system (Venkatasubramanian, 1985; Venkatasubramanian, 1986; Gani and O'Connell, 1989). This approach is now more viable

because expert system techniques are becoming more widely recognized. When these techniques were unknown, except within the computer science community, the knowledge engineer was required. Now that shells are available and expert system techniques are well documented, most domain experts are capable of building their own expert systems by

a) adding domain knowledge to expert system shells, or

b) building expert systems.

The second problem in expert system development is that current expert systems rely primarily on symbolic information. Engineering is characterized by a mixture of both numeric and symbolic information. For example, the design of a piece of equipment requires several passes. The first is the conceptual design stage and involves the application of rules of thumb to obtain a general design. The next pass refines the design further by introducing more specific numerical information. A final pass is done to ensure a safe design that is up to specification. Each of the above passes may require several attempts depending on the complexity of the equipment. One approach is to use expert systems within problem areas where only symbolic information is involved. This will limit the application of expert system technology. Another is to use coupled expert systems, combined expert and algorithmic systems, which are capable of performing both numeric and symbolic processing. This can be achieved in one of two ways. The first is to use a meta-system which coordinates the separate programs, one for symbolic and another for numeric processing (Rao, Jiang and Tsai, 1989). The second approach is to combine both types of information processing within the same program.

In this study, two expert system prototypes were developed to investigate issues with expert system development in chemical engineering. The first expert system is designed to provide advice for the selection of vapor-liquid equilibrium calculation (VLEC) methods given a binary set of chemicals. The problem of capturing the knowledge to select VLEC methods was solved using machine learning. Through machine learning, it was possible to

relate certainty factors for the knowledge base rules to the domain knowledge. Numeric and symbolic processing was performed within one system and was developed in a flexible object-oriented programming language.

The second expert system was developed in conjunction with an industrial partner. Its purpose is to provide advice for the startup of a section of a refinery. Knowledge for this system was obtained through interview of experts, a good set of written documentation and through personal hands-on experience. It was built using an expert system shell.

1.2 Organization

This thesis consists of seven chapters. The first is this introductory chapter, which covers the general objectives and organization of the thesis. The second chapter provides background information about artificial intelligence and expert systems. The artificial intelligence terminology used in this thesis is defined. The thesis research was based on primarily on two case studies. The first is covered in chapter 3. This case study involves building an expert system for the selection of vapor-liquid equilibrium calculation methods. The second case study, described in chapter 4, is an expert advisory system to provide advice during the startup of a section of a refinery. Chapter 5 contains a general discussion of the conclusions for the two case studies. The specific conclusions related to each case study are found at the end of their respective chapters. References for this thesis are listed in chapter 6. The last chapter contains all the appendices for the thesis.

2. Background on Expert Systems

This chapter introduces the reader to the basics of artificial intelligence, expert systems and machine learning. This is not an exhaustive description, but covers the concepts and definitions used in the two case studies.

2.1 Definitions

The following sections present the basic definitions for the terms used in this thesis. These definitions are not strict since the study of artificial intelligence is not mature.

2.1.1 Artificial Intelligence

Artificial intelligence (AI) is a field of study that deals with the creation of intelligent machines (Lazure, 1988; Luger and Stubblefield, 1989). Intelligent machines perform tasks that are considered complex and require significant expertise when performed by a human. It should be noted that a rigorous definition of AI has not been agreed upon. Expert systems development is one branch of AI.

2.1.2 Expert Systems

An expert system can be defined as a computer program which contains the knowledge of a specific domain and uses this knowledge to solve problems within the domain (Lazure, 1988; Luger and Stubblefield, 1989). Simulation of an expert's problem-solving

ability is the main objective of expert system development. The method of solution is usually based on a search technique called inferencing. The inference engine searches through the knowledge base until enough knowledge has been extracted to solve the problem. An expert system can be further characterized by the use of certain organizational structures. This structure is the separation of the knowledge from the inference mechanism and the use of symbolic reasoning as opposed to numeric computation to solve problems.

There are four main parts to an expert system, as shown in Figure 2.1. The first is the knowledge base. It contains the identified knowledge about the domain. The knowledge base is the key feature that distinguishes expert systems from other computer programs. It is also the most complex part to build. The second part is the inference engine. Its function is to search the knowledge in some organized fashion for knowledge appropriate to the problem at hand. The third part is the working memory where a description of the problem, the solution and any intermediate information is stored. The final part is the user interface.

In summary, an expert system can be defined by the following characteristics:

1) emulates an expert's problem-solving ability,
2) possesses knowledge about a narrow domain,
3) the knowledge base is stored separately from the inference engine,
4) the knowledge is represented in symbolic form,
5) an inference engine searches the knowledge for solutions to the given problem,
6) uncertain information can be handled.

Expert system technology is a new technique and is not used to replace current techniques of problem solving. It supplements current technology so problem solving can become more efficient. There are several benefits of expert systems. Knowledge and experience can be distributed to non-experts with 24 hour availability. Updating and maintenance of the expert

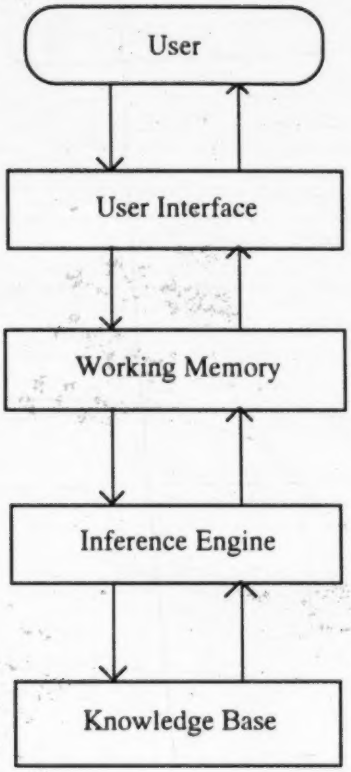


Figure 2.1 Structure of an Expert System

system results in accumulation of several expert's knowledge. Another important use is to capture and retain an expert's knowledge so that his retirement is not as detrimental to the company.

2.1.3 Knowledge Representation

Knowledge representation is a key issue in the construction of the knowledge base. Production rules are widely used to represent knowledge. A production rule is a condition-conclusion pair which defines a single element of problem-solving knowledge (Luger and Stubblefield, 1989). A typical format for a production rule is given as the following,

IF condition(s) with certainty CF1
THEN conclusion(s) with certainty CF2

where CF1 and CF2 denote certainty factors.

The rules represent symbolic relationships between attributes and goal states. A goal state is one possible solution to some problem within the domain. Attributes are descriptions of the problem state. Certainty factors, CFi, are used to describe the degree of confidence of some piece of knowledge. Certainty factors are also known as confidence factors (Luger and Stubblefield, 1989). Sometimes the attributes of the problem are not known with full certainty. For instance, a tank's liquid level is thought to be normal but only with a 75% degree of certainty. In this case, the attribute describing the tank's liquid level would have a value of 'normal' with a certainty of 75% and would be represented as,

tank_level = normal
CF = 0.75

Another use of certainty factors is for the relationship between attributes and goal states. During knowledge acquisition, each rule is assigned a degree of confidence. It represents the degree of certainty that the conclusion is correct given the current conditions. The certainty factor may be assigned an absolute value or it may be calculated based on statistical theory. The certainty factor approach is difficult to implement because the degree of uncertainty is not obvious. Their implementation is too subjective and prone to errors. A more rigorous method of relating the degree of certainty to the domain knowledge is needed.

2.1.4 Inference Engine

Inferencing is a search technique through a set of production rules (Luger and Stubblefield, 1989). The search process is used to identify knowledge relating to the problem being considered. The most common forms of inferencing are forward-chaining and backward-chaining.

Forward-chaining uses the input attributes to identify possible goals. This is the common approach to IF..THEN rules. The attributes are tested in the condition statement and the goals are produced in the conclusion statement. Backward-chaining is the opposite to forward-chaining. It uses the goal states to produce attributes. These attributes are compared with those actually present to determine if the current goal is valid. The goal state is selected from the conclusion statement and then the attributes are produced from the condition statement.

Each of these chaining methods are further modified by either depth-first or breadth-first search. During the inferencing process attributes may get modified or new attributes may be identified. These changes of state are caused by the firing of rules. In forward-chaining,

a rule fires when the condition statement is true and the conclusion statement is executed. A depth-first search uses all previous modifications to the attributes caused by firing rules. After each rule is fired, a search is conducted for the next rule which may be fired. Once identified, it is fired and the new attribute values are used to continue the search. A breadth-first search does not allow a firing rule to change the attributes until it is determined that no other rule can currently be fired. Each rule that could be fired, based on the current attribute values, is fired to see if it produces a final goal. If it does not, then the attributes are returned to their state before firing the rule. It then searches for another rule which could be fired. If no other rules can be fired then one of the current rules is fired to modify the attributes and the breadth-first search is repeated.

Heuristics are used to make the inference process more efficient (Luger and Stubblefield, 1989). Heuristics are knowledge which may increase the efficiency of solving a problem (Lazure, 1988). They do not guarantee a solution or increase in efficiency. If the knowledge space is very large then heuristics can be used to identify areas where search should continue. Efficiency is improved by not performing an exhaustive search of the knowledge. Inference algorithms can be developed entirely from heuristics or as a combination of heuristics and chaining techniques.

During the inference process it is common to encounter two or more rules which are currently capable of being fired. These rules are said to be in conflict. The selection of which rule to fire is called conflict resolution (Luger and Stubblefield, 1989). Conflict resolution can be based on heuristics, certainty factor or rule priorities. During the construction of the knowledge base, rules can be assigned priorities so that certain rules take precedence over others. Certainty factors can be used to fire the rule with the highest or lowest certainty.

Evaluation functions are used to reduce the search time in large knowledge bases (Luger and Stubblefield, 1989). They can take a number of forms but they usually involve

heuristics (Lazure, 1988). Evaluation functions convert symbolic information into numeric form. The result is that search criteria can be ranked in numerical order. Selection of the highest ranking search path is known as a best-first search (Lazure, 1988).

2.2 Machine Learning

Machine learning is a technique used to provide an intelligent system the ability to acquire knowledge. Learning is defined as changes in knowledge that allows the system to more efficiently or effectively solve a problem (Shavlik and Dietterich, 1990).

Learning is implemented by an algorithm that extracts information from domain examples. These algorithms are usually domain specific although some generalized techniques have been developed (see Shavlik and Dietterich, 1990). Learning algorithms can be divided into several categories. The first is supervised versus unsupervised learning. In unsupervised learning the expert system extracts knowledge from each real consultation session. Supervised learning restricts the training examples to those selected by a human supervisor (Shavlik and Dietterich, 1990).

Another distinguishing characteristic is whether or not the system displays knowledge level learning (Dietterich, 1986). Knowledge level learning occurs when changes in the system result in new problem solutions that were not previously present. Non-knowledge level learning is when a system reorganizes its knowledge into a more efficient structure. This new structure may result in faster inference times or require less user volunteered information but the same solutions are reached.

Noise can be a source of complication for machine learning. Noise is defined as errors in the training examples (Kibler and Langley, 1988). These errors are propagated into the

knowledge base through the learning process. The effect of noise can be reduced by relying on supervised learning. The degree of noise reduction depends on the quality of the screening process by the supervisor.

Many machine learning algorithms perform knowledge classification. These are known as classifier systems. The system begins with a known set of classifications that describe the domain. Training examples are provided and the system uses the classifications to build descriptions of the examples. These descriptions form the knowledge base. Consultation of classifier systems begin with identification of the pertinent classifications. The knowledge that matches these classifications form the basis for the problem solution.

3. Case Study One: A Learning Expert System for Vapor-Liquid Equilibrium Calculations*

This chapter describes an expert system for the selection of vapor-liquid equilibrium calculation methods. The system implements machine learning to acquire knowledge.

3.1 Introduction

The area of chemical thermodynamics is very complex and diverse. The particular aspect of thermodynamics studied here is the prediction of vapor-liquid equilibrium (VLE) data. Prediction of VLE data is performed by using one of many calculation algorithms and methods. Each method is better suited under different conditions and with different chemical compounds. The problem is the proper selection of a VLE calculation (VLEC) method given the chemicals and conditions involved.

An illustration of the difficulty in selecting VLEC methods was supplied by Angel, Marmur and Kehat (1986). They compared several popular VLEC methods to determine relative accuracies. For the light hydrocarbon example, the method that best predicted the split components was the UNIFAC method. This is surprising because the Peng-Robinson equation of state is typically recommended for light hydrocarbon systems. It is cases like these that make the selection of VLEC methods so complex.

VLE data is important for both design and simulation of various processes. Stripping, absorption and distillation are all based on the principle of VLE. Quite arguably, actual experimental VLE data is the best to use. However, experimental data may not be available or it may be too expensive to obtain. The alternative is to use one of the many prediction schemes available.

---

* A version of this chapter has been presented at the 1990 Summer Computer Simulation Conference, July 16-18, 1990 in Calgary, Alberta, Canada. Ref: **Dong, R.** and M. Rao. (1990). "A Learning Expert System for Phase Equilibrium Prediction Method Selection." Proc. of the 1990 Summer Computer Simulation Conference, pp. 380-382.

The objective of this project was to investigate the use of machine learning for a VLE expert system. The project is not aimed at determining the best VLEC methods. Nor is it aimed at predicting the most accurate VLE data. The project is aimed at developing an expert system capable of learning from experimental data and making recommendations based on its acquired experience. Since there is a large amount of VLE data available, more than any one person could analyse and assimilate, a method should be found which efficiently identifies appropriate VLEC models using a small subset of the VLE data. This project investigates the use of machine learning as a tool to perform this identification.

3.2 Background

There has been previous research in building expert systems for VLE prediction. This section begins with a description of the previous work done by others. Another section describes some of the thermodynamic theory involved with the present work.

3.2.1 Previous Research in Expert Systems for Thermodynamics

Banares-Alcantara, Westerberg and Rychener developed an expert system for VLEC method selection called CONPHYDE (Banares-Alcantara, Westerberg and Rychener, 1985). It was built using the framework of PROSPECTOR, an early expert system used for locating mineral deposits. CONPHYDE prompts the user for pertinent information through questions which are answered with a format of yes/no and certainty factor. Based on the input by the user, CONPHYDE makes recommendations for a VLEC method. Input data includes the compounds present, the operating conditions and any VLEC method preferences. One conclusion by these authors is that engineering expert systems will be required to perform both symbolic and numeric processing. CONPHYDE did not, however, perform numeric processing. The development of CONPHYDE was done using the traditional knowledge

engineering approach.

Venkatasubramanian also developed an expert system for VLEC method selection (Venkatasubramanian, 1985), but instead of using the traditional approach of CONPHYDE he used a stochastic parallel network. The network arrived at a conclusion by minimizing a cost function subject to the constraints. Input data consisted of properties of the fluid mixture, operating conditions, etc. The requirement to identify the fluid mixture properties before consultation was a draw back. For this system, development was done by a domain expert without assistance from a knowledge engineer.

Gani and O'Connell built a knowledge based system for selecting physical property and phase equilibrium methods for process design and simulation (Gani and O'Connell, 1989). It was based on a set of indices and matrices which were traversed during consultation. Very little direct questioning of the user was done because of the reliance on internal databases. A benefit of this system was the type of thermodynamic calculation is considered as an input. Additional inputs were the compounds, operating conditions and the range of compositions involved. A further refinement was a specification of which phase is of interest. This system was also developed by domain experts.

All of the above systems were developed using the knowledge of one or two domain experts. Any new knowledge that is to be added to the system must be supplied by an expert. Relying on a few experts may not be appropriate for the area of thermodynamics. Even the limited portion involved with VLE prediction is very complex. There are thousands of compounds and dozens of VLE methods. The combinations are numerous. There are many new VLEC methods being proposed in the literature. It is a difficult task for any expert to be aware of and to evaluate all the new methods. Machine learning is proposed as a solution to this shortcoming. It would accelerate the evaluation of new methods and ensure a wide variety of tests.

3.2.2 Vapor-Liquid Equilibrium

Phase equilibrium occurs when two phases, such as liquid and vapor, are in contact and have no net mass or energy transfer. An example is a mixture of ice and water. More important is the phase equilibrium achieved when mixtures of components are in both phases. Mixture effects due to molecular interactions can cause wide variations in behavior. These variations make phase equilibrium very complex. However, it is mixtures and their behavior that most concern chemical engineers and vapor-liquid equilibrium. VLE is used to model and design distillation towers, flash drums, absorption/stripping towers, two-phase flow, etc.

The mathematical description of phase equilibrium states that the temperature, pressure and chemical potential for each component is the same in all phases. The most common parameter which is used to describe equilibrium is the ratio of a component's composition in both phases. It is called the equilibrium ratio. For vapor-liquid equilibrium, it is defined as (Kyle, 1984),

$$K_i = \frac{y_i}{x_i} \qquad (3.1)$$

The compositions are given as mole fractions. A good summary of references for fluid phase equilibrium is given in Reid, Prausnitz and Poling (1987).

Numerous techniques have been developed to predict the equilibrium ratio for various compounds at different operating conditions. A common technique is equations of state. Equations of state are mathematical relations describing the mutual behavior of pressure, temperature and volume for a given compound. In this investigation only a few of the cubic equations of state are involved. Cubic equations of state are relations which can

be algebraically manipulated into a cubic polynomial form with respect to volume or compressibility factor. The compressibility factor is defined as,

$$Z = \frac{PV}{RT} \qquad (3.2)$$

Cubic equations of state are popular because they provide reasonably accurate results and fast calculations. However, for strongly interacting molecules the accuracy of the results may be poor. The cubic equations of state that were included and compared in this study were the following:

1) Redlich-Kwong
2) Soave-Redlich-Kwong
3) Peng-Robinson.

Each of these equations can be given as a specialized case of a general form which is applicable to many cubic equations of state (Reid, Prausnitz and Poling, 1987),

$$P = \frac{RT}{(V-b)} - \frac{a}{(V^2+ubV+wb^2)} \qquad (3.3)$$

Table 3.1 gives a complete description of these cubic equations of state. An alternative form is the cubic polynomial in compressibility,

$$Z^3 + (uB-1-B)*Z^2 + (A+wB^2-uB-uB^2)*Z + (-AB-wB^2-wB^3) \quad 0 \qquad (3.4)$$

$$A = \frac{aP}{R^2T^2} \qquad B = \frac{bP}{RT}$$

Table 3.1 Three Cubic Equations of State

Equation	u	w	b	a
Redlich-Kwong	1	0	$\frac{0.08664RT_c}{P_c}$	$\frac{0.42748R^2T_c^{2.5}}{T^{1/2}P_c}$
Soave-Redlich-Kwong	1	0	$\frac{0.08664RT_c}{P_c}$	$\frac{0.42748R^2T_c^{2}}{P_c}[1+f(\omega)(1-T_r^{1/2})]^2$ where $f(\omega)=0.48+1.57\omega-0.176\omega^2$
Peng-Robinson	2	-1	$\frac{0.0778RT_c}{P_c}$	$\frac{0.45724R^2T_c^{2}}{P_c}[1+f(\omega)(1-T_r^{1/2})]^2$ where $f(\omega)=0.37464+1.5422\omega-0.26992\omega^2$

(Ref: Reid, Prausnitz and Poling, 1987)

This form is for pure components. Mixtures require some special handling.

Mixing rules have been developed to modify the parameters in the equations of state so mixture behavior is more closely predicted. For the above cubic equations the mixing rules are presented as follows (Reid, Prausnitz and Poling, 1987),

$$a_m = \sum_i \sum_j y_i y_j (a_i a_j)^{1/2} (1 - k_{ij}) \qquad (3.5)$$

$$b_m = \sum_i y_i b_i$$

The equation of state parameters, 'a' and 'b', are taken from the pure component form given in Table 3.1. The interaction parameter, k_{ij}, is an empirical correlation factor used to adjust the mixing rules. It is needed because the mixing rules and the equation of state do not perfectly predict the mixture's behavior. For light hydrocarbon systems it is common to ignore k_{ij} (Reid, Prausnitz and Poling, 1987) because cubic equations closely predict their behavior. Equations of state are designed to account for molecular interactions and molecular volume. In this study the interaction parameters were not included because data was not available for most of the training sets. Furthermore, including interaction parameters would not alter the expert system framework being built in this study. The mixing rules, for k_{ij} = 0, are reduced to,

$$a_m = \sum_i \sum_j y_i y_j (a_i a_j)^{1/2} \qquad (3.6)$$

$$b_m = \sum_i y_i b_i$$

The use of cubic equations of state to predict equilibrium ratios is based on the theory of fugacity. Fugacity is a thermodynamic property that is related to pressure (Kyle, 1984).

The exact relation of interest is the fugacity coefficient which is defined as,

$$\phi = \frac{f}{yP} \tag{3.7}$$

The fugacity coefficient can be predicted using the following relation derived from the definition of the fugacity coefficient and the general cubic equation of state (Reid, Prausnitz and Poling, 1987),

$$\ln \phi = \frac{b_i}{b_m}(Z-1) - \ln(Z-B) + \frac{A(\frac{b_i}{b_m} - \delta_i)}{B(u^2 - 4w)^{1/2}} \ln\left[\frac{2Z + B(u+(u^2 - 4w))^{1/2}}{2Z + B(u-(u^2 - 4w))^{1/2}}\right] \tag{3.8}$$

$$\delta_i = 2(\frac{a_i}{a_m})^{1/2}$$

This relation is used to obtain the fugacity coefficient for both phases. The equilibrium ratio is then obtained as,

$$K = \frac{\phi_L}{\phi_V} \tag{3.9}$$

Another popular method of predicting VLE is activity coefficients. These methods are used to model interactions in the liquid phase. The main equation to calculate the equilibrium ratio is the following (Kyle, 1984),

$$K_i = \frac{P_{vp}\gamma_i}{P\phi_V} \tag{3.10}$$

When using activity coefficient methods the vapor phase fugacity coefficient is usually assumed to be unity. The vapor pressure is obtained from correlations. In this system, the Wagner equation (Reid, Prausnitz and Poling, 1987) is used to predict the vapor pressure,

$$\ln \frac{P_{vp}}{P_C} = \frac{a\tau + b\tau^{1.5} + c\tau^3 + d\tau^6}{T_r} \qquad (3.11)$$

$$\tau = 1 - T_r$$

Three activity coefficient methods are implemented here. These are the following,

1) Wilson equation
2) van Laar equation
3) Margules equation.

Table 3.2 lists the equations and their parameters (Reid, Prausnitz and Poling, 1987). Each method requires the specification of two experimental parameters. These parameters are calculated by fitting the equations' predictions to experimental data. These parameters must be supplied before the method is used. The equations of state have molecular interactions and volumes built into the equation structure. For this reason, they are still useful when there are no interaction parameters available for them. The activity coefficient equations are heavily reliant on these parameters. The reason for this is that the activity coefficient methods are designed to model strong interactions between molecules.

Another VLEC method variation is to combine the activity coefficient and the equation of state methods. The activity coefficient method is used to model the liquid phase. The vapor phase is modelled by the equation of state. The vapor phase fugacity coefficient is calculated by an equation of state instead of being set to unity. The parameters are obtained

Table 3.2 Activity Coefficient Equations

Name	Equations	Note: A_{12}, A_{21} are experimental parameters

Wilson

$$\ln \gamma_1 = - \ln (x_1 + \Lambda_{12} x_2) + x_2 \left[\frac{\Lambda_{12}}{x_1 + \Lambda_{12} x_2} - \frac{\Lambda_{21}}{\Lambda_{21} x_1 + x_2} \right]$$

$$\ln \gamma_2 = - \ln (x_1 \Lambda_{21} + x_2) - x_1 \left[\frac{\Lambda_{12}}{x_1 + \Lambda_{12} x_2} - \frac{\Lambda_{21}}{\Lambda_{21} x_1 + x_2} \right]$$

$$\Lambda_{ij} = \frac{v_{jL}}{v_{iL}} \exp \left[- \frac{A_{ij}}{RT} \right]$$

Van Laar

$$\ln \gamma_1 = \frac{A_{12}}{RT \left[1 + \frac{A_{12} x_1}{A_{21} x_2} \right]^2} \qquad \ln \gamma_2 = \frac{A_{21}}{RT \left[1 + \frac{A_{21} x_2}{A_{12} x_1} \right]^2}$$

Margules

$$\ln \gamma_1 = \frac{(A_{12} + 3A_{21}) x_2^2 - 4A_{21} x_2^3}{RT}$$

$$\ln \gamma_2 = \frac{(A_{12} - 3A_{21}) x_1^2 + 4A_{21} x_1^3}{RT}$$

(Ref: Reid, Prausnitz and Poling, 1987)

using the same methods described earlier. All the combinations of the three equations of state and three activity coefficient methods were included in this system.

The VLEC methods are evaluated using a bubble-point calculation. The algorithm for the calculation is shown in Figure 3.1, (Henley and Seader, 1981). The calculation produces an estimate for the vapor phase composition and temperature given the pressure and liquid phase composition. The estimates can be directly compared to experimental data to test accuracy.

3.3 System Description

The previous section built a basic understanding of VLE fundamentals. This section presents a detailed description of the developed expert system.

3.3.1 Chemical Classification

To develop an expert system for VLEC method selection the chemical compounds were separated into chemical families. This technique was implemented in the rule-based expert systems presented in the section 3.2.1. The difficulty here is to determine a good set of classification rules. Classifications are often based on some common property. In this system the common properties are related to the determination of VLE data. The classification rules implemented here are similar to the system used in CONPHYDE (Banares-Alcantara, et al. 1985). The classification families are listed in Figure 3.2, and are separated into three distinct levels.

The first level is the collection of all chemical compounds. Level 2 divides Level 1 into three groups. Each of the Level 2 groups are divided into several families. Level 3 has

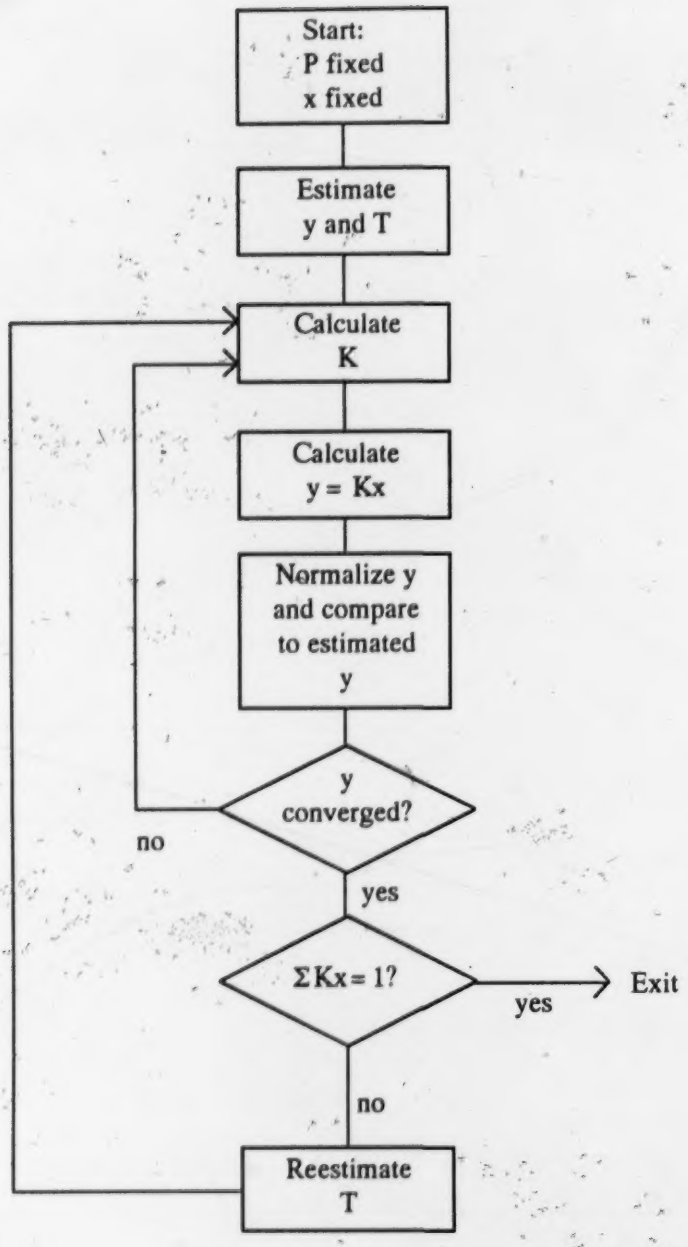


Figure 3.1 Bubble-Point Calculation Algorithm

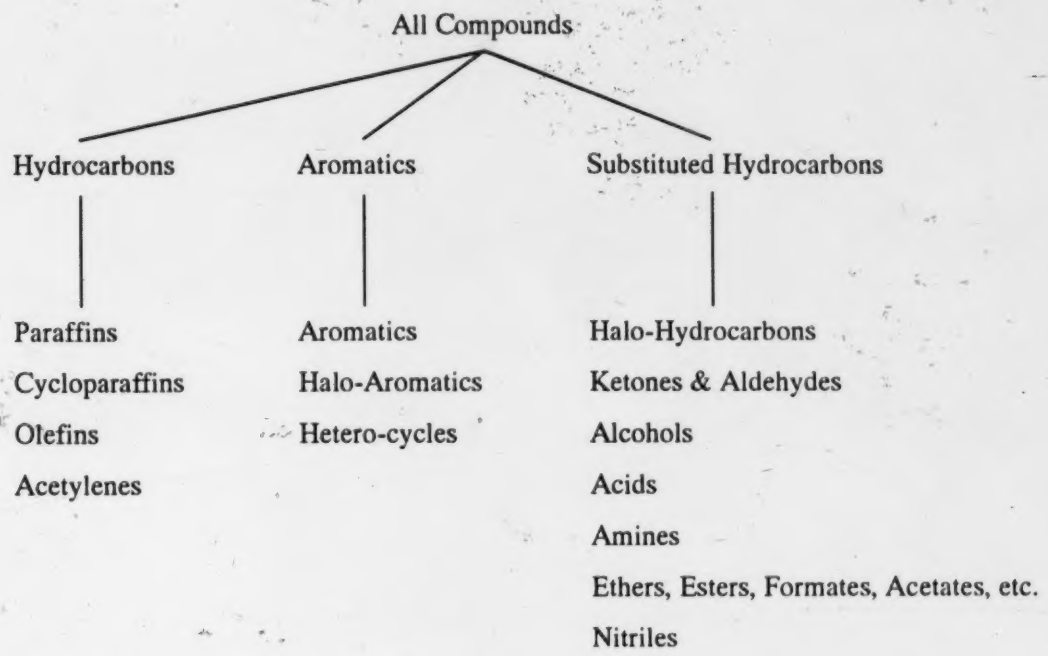


Figure 3.2 Classification Families for Chemical Compounds

the most division of the domain and therefore, captures the most detail. A level that is present but not shown is the actual compounds. These could be viewed as Level 4. This level involves no generalization and is not suited for symbolic processing. The rules for this classification system were implemented in the standard production rule format. The classification knowledge base is contained in Appendix B.

The degree of knowledge division is a key issue that determines the efficiency of the expert system. Smaller divisions of knowledge result in greater detail and a closer representation of reality. However, this greater detail reduces the system's ability to make deductive conclusions. An expert system should have several levels of generalization available. The inference process should try to use the most specific knowledge first. Being more specific should improve the prediction. If this knowledge fails or is not available then more general knowledge should be used. The three classification levels described here are implemented in this way.

3.3.2 Machine Learning

Learning systems are usually trained with examples. In my system, experimental VLE data are presented as learning examples. As with other learning systems, this system modifies the knowledge base to match the training data as close as possible.

Each experimental data point contained values for the properties shown in Table 3.3. The learning functions used the information to perform a bubble point calculation. Each VLE method was used to determine the vapor composition and temperature at equilibrium. These predicted values were compared to the experimental values. The errors are a measure of each VLE method's predictive quality for the given components. The objective of the learning system is to acquire some value representing the predictive quality of each VLE method. The errors are the natural values to use as an evaluation for each VLE method. The errors in this

Table 3.3 Experimental Data

Compound ID's

Temperature

Pressure

Liquid Composition

Vapor Composition

Wilson Equation Parameters

Van Laar Equation Parameters

Margules Equation Parameters

system are stored separately. One is the error in vapor composition. Another is the error in temperature.

The errors in VLE prediction are the standard means of evaluating these methods. It is not desirable to store the errors for each set of components and VLE methods. This would become a database without any ability to generalize from its information. Another drawback is the amount of data that would require storing. The combinations of the thousands of compounds with the dozens of VLE methods would require substantial computer storage. An alternative form of storing the errors without resorting to the database approach must be found. In AI terminology, this is known as selecting the knowledge representation scheme.

3.3.3 Knowledge Representation

One knowledge representation approach, errors are classified into a symbolic space ranging from poor to excellent. For example,

poor	> 100% error
fair	> 50% & < 100%
good	> 20% & < 50%
very good	> 10% & < 20%
excellent	< 10%.

The problem with this approach is how to continue accumulating information as new training examples are presented. An example is how to characterize a method which provided the following range of estimates?

1 case of fair (97% error)

2 cases of good (48% and 44%)

1 case of very good (15%).

Using symbolic reasoning, the above method would be identified as being usually Good or better. This type of information is not very useful to an engineer trying to solve this type of problem. A better approach is to maintain a running average of the errors encountered in the training sets. The above example would give an average or expected error of 51%, which is Fair by the above classification system.

$$\text{Ave. Error} = \frac{(\text{Old Ave. Error})*(\text{Old Number of Cases}) + (\text{Current Error Value})}{(\text{Old Number of Cases}) + 1} \qquad (3.12)$$

Another problem with the symbolic grading method is how to define the ranges of error. An engineer will have different tolerances of error depending on the type of problem he or she is solving. In some cases nothing more than 5% error is tolerable, while others may allow up to 50% or more. The selection of a symbolic error limit is related to the numeric ranges assigned to the grades. This defeats the purpose of the symbolic grading approach.

The approach taken in this expert system was to maintain accumulating average absolute errors. These average errors are kept for each binary pair of chemical families. To maintain the average error a running count of the number of training sets for each chemical family pair was stored. The formula for maintaining the average errors is as follows,

The average, or expected, errors can be viewed as a type of confidence factor. Lower expected errors mean higher confidence in the VLE method. This results in a set of rules of the following form,

IF (mixture is an Alcohol and Paraffin)
THEN (use Wilson activity coefficient method,
Expected Error in Composition = xx,
Expected Error in Temperature = xx)

IF (mixture is an Alcohol and Olefin)
THEN (use Wilson activity coefficient method,
Expected Error in Composition = xx,
Expected Error in Temperature = xx)

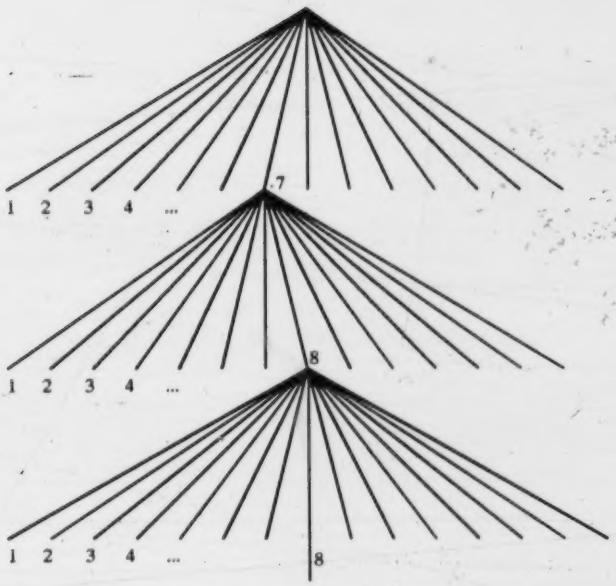
etc.

IF (mixture is an Alcohol and Paraffin)
THEN (use Peng-Robinson equation of state,
Expected Error in Composition = xx,
Expected Error in Temperature = xx)

IF (mixture is an Alcohol and Olefin)
THEN (use Peng-Robinson equation of state,
Expected Error in Composition = xx,
Expected Error in Temperature = xx)

etc.

There would be one rule for each pair of families. There would be one complete set of chemical family rules for each VLE method. The above method would produce 1575 rules for Level 3. Figure 3.3 illustrates the geometric growth caused by the combinations of chemical families and VLE methods. Using this format was not feasible, so the rules were



Classify Compound 1 into one of 14 Families

Classify Compound 2 into one of 14 Families

Rules for the 15 VLE methods

IF mixture is Family 7 and Family 8

THEN use VLE method 8

Figure 3.3 Geometric Growth of the Production Rules

restructured. By taking advantage of the regular pattern of the rules, the following structure was obtained,

IF (mixture is an Alcohol and Paraffin)
THEN (recommended VLE method = EF (Alcohol, Paraffin))

IF (mixture is an Alcohol and Olefin)
THEN (recommended VLE method = EF (Alcohol, Olefin))

where EF = evaluation function, returns the VLE method with the lowest expected error for the given binary mixture.

The evaluation function performs a search through the error matrices until it locates the one with the lowest error. This format results in a knowledge base containing 105 rules for Level 3. For all three levels of compound classification there are 112 rules. These are shown in Appendix B.

3.3.4 Inferencing

Inferencing in this system is based on a heuristic search. There are two levels of rules. The first is concerned with the classification of the compounds involved. The second level determines which VLEC method to select. From this viewpoint, the matrices of expected errors are a measure of confidence used for conflict resolution. The VLEC method with the highest confidence, or lowest error, will be recommended. In the case of a consultation for a pair of chemical families for which there is no training data, the system is unable to recommend a VLEC method. To solve this problem the inference engine moves to a higher classification level. In this way, training data for related families can be used as the basis for

selection. The inference process can be summarized as follows:

1) Obtain the compounds' ID numbers.

2) Classify each compound.

- search classification KB for a rule to classify each of the compounds
- classification is done for both Level 2 and 3

3) Begin search at Level 3.

- evaluation function selects VLEC with lowest error in composition
- evaluation function selects VLEC with lowest error in temperature
- recommendations are made to user
- if no training data has been presented for this pair of families then go to Level 2.

4) If no suitable selections from Level 3 then use Level 2.

- evaluation function selects VLEC with lowest error in composition
- evaluation function selects VLEC with lowest error in temperature
- recommendations are made to user
- if no training data has been presented for this pair of families then go to Level 1.

5) If no suitable selections from Level 3 or 2 then use Level 1.

- evaluation function selects VLEC with lowest error in composition
- evaluation function selects VLEC with lowest error in temperature
- recommendations are made to user
- if no training data has been presented then no recommendation can be made.

6) End inferencing.

3.3.5 System Structure

The knowledge in this expert system was grouped into three databases (DB) and two knowledge bases (KB). The databases consisted of the pure component DB, VLE calculation method DB and the training set DB. The KB's contained the necessary information for the selection of one or more VLE methods given the type of compounds involved. Figure 3.4 shows the grouping of the DB's and KB into one entity called the complex information base (CIB).

The pure component DB contained the necessary information to characterize and distinguish one component from another. Table 3.4 shows a list of the component properties stored in the DB. Not all these properties are needed when performing a VLE calculation. There are 141 compounds in the pure component DB and they are listed in Appendix A.

VLE calculation methods were stored in a separate DB. Although each method was compiled and linked into the source code, direct access to the methods was restricted. An alternative was to separately compile each method and store its executable name in a list file. The expert system could then look in the list file for each executable. This technique would simplify the addition of new VLE methods because a complete re-linking of all the source is not necessary. Each VLE calculation method needed pure component properties as input parameters as well as experimental data that it was compared with. The resulting errors in the prediction are returned. There are 15 VLE methods in this DB and they are listed in Table 3.5.

The third DB stores all the training examples that were presented to the system. This DB is important for the evaluation of new VLE methods as they are added to the system. Once new VLE methods are added, they can be trained using the same data sets. Currently, there are over 400 experimental data points stored in this DB.

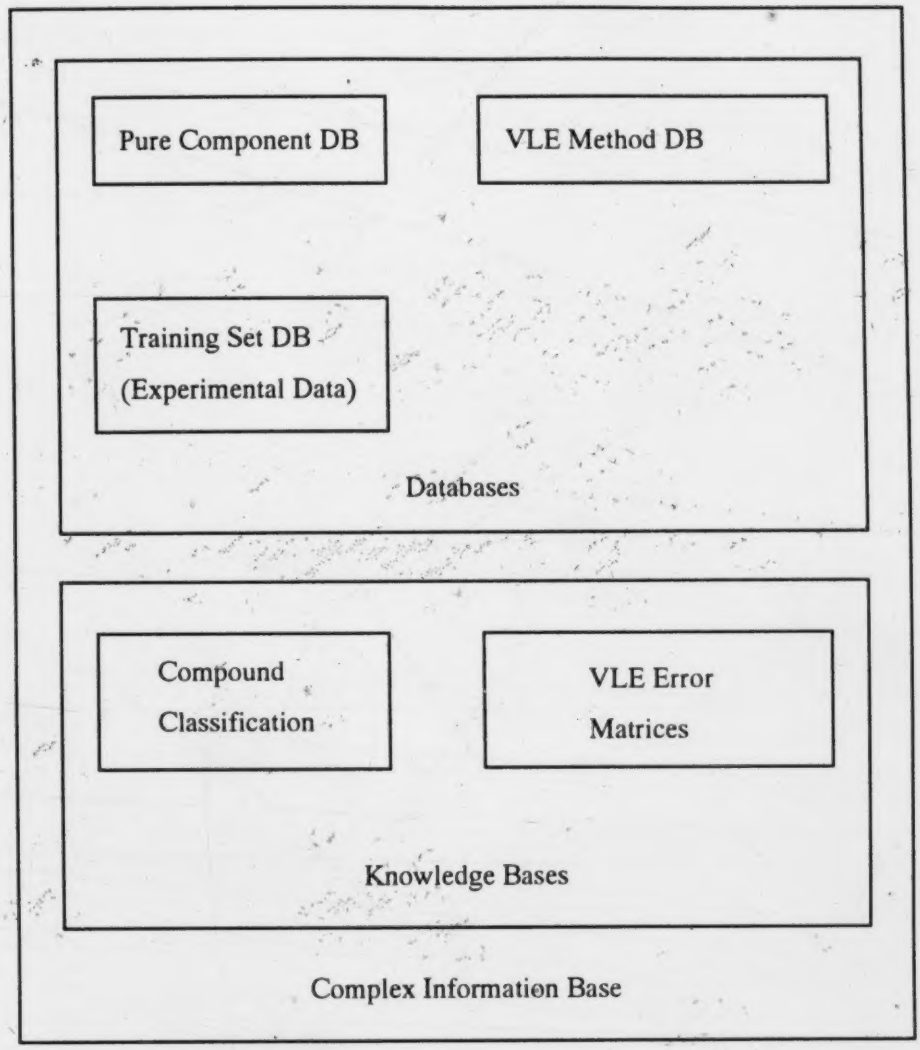


Figure 3.4 Complex Information Base

Table 3.4 Pure Component Properties

Molecular Formula

Name(s)

Molecular Weight

Critical Temperature

Critical Pressure

Critical Volume

Critical Compressibility

Acentric Factor

Vapor Pressure Corelation

Heat Capacity Correlation

Liquid Density Measurement

Temperature of Liquid Density Measurement

Table 3.5 Listing of the 15 VLEC Methods in this System

VLE Method #	Liquid Phase	Vapor Phase
1	Peng-Robinson	Peng-Robinson
2	Redlich-Kwong	Redlich-Kwong
3	Soave-Redlich-Kwong	Soave-Redlich-Kwong
4	Wilson	None
5	Wilson	Peng-Robinson
6	Wilson	Redlich-Kwong
7	Wilson	Soave-Redlich-Kwong
8	Van Laar	None
9	Van Laar	Peng-Robinson
10	Van Laar	Redlich-Kwong
11	Van Laar	Soave-Redlich-Kwong
12	Margules	None
13	Margules	Peng-Robinson
14	Margules	Redlich-Kwong
15	Margules	Soave-Redlich-Kwong

The KB had two sections. The first is the compound classification rules. The second contained the certainty factors for each VLE method. There were three levels of classification of the compounds. These are shown in Figure 3.5. Level 1 is a classification of all chemical compounds. Level 2 has three categories: hydrocarbons, aromatics, substituted hydrocarbons. Level 3 is broken into 14 categories. The error matrices for the Peng-Robinson equation of state for each classification level are shown in Figures 3.6 to 3.8. The figures indicate that each cell contains a rule. Only the certainty factors were stored in the matrices. The knowledge base is reproduced in Appendix B. There are 18 rules in the classification KB. There are 105 rules for Level 3, 6 rules for Level 2 and 1 rule for Level 1. This makes for a total of 130 rules.

3.3.6 Implementation Platform

The system was written in C++ using the object-oriented programming paradigm (Stefik and Bobrow, 1986; Stroustrup, 1986). This assisted in the integration of numeric and symbolic processing. C++ is a very versatile language which is capable of emulating many of the features of PROLOG or LISP while retaining superior flexibility. The success of C++ in artificial intelligence programming will be dependent on the creation of a class library that provides the benefits of PROLOG or LISP. The classes built for this system are not complete enough to represent this desired class library but were sufficient for this problem. Figure 3.9 presents the classes built for this system. The source code is contained in Appendix C.

3.4 Evaluation

This section presents how the system works through several examples. The examples illustrate the learning process. Inferencing is described graphically. Evaluation of the system

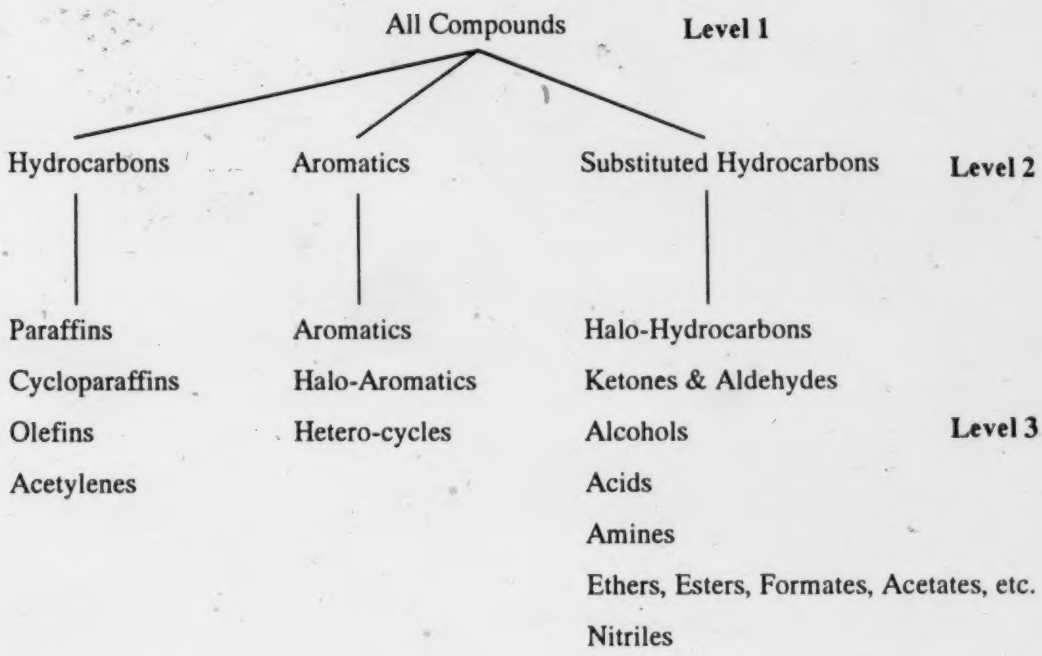


Figure 3.5 Classification Levels

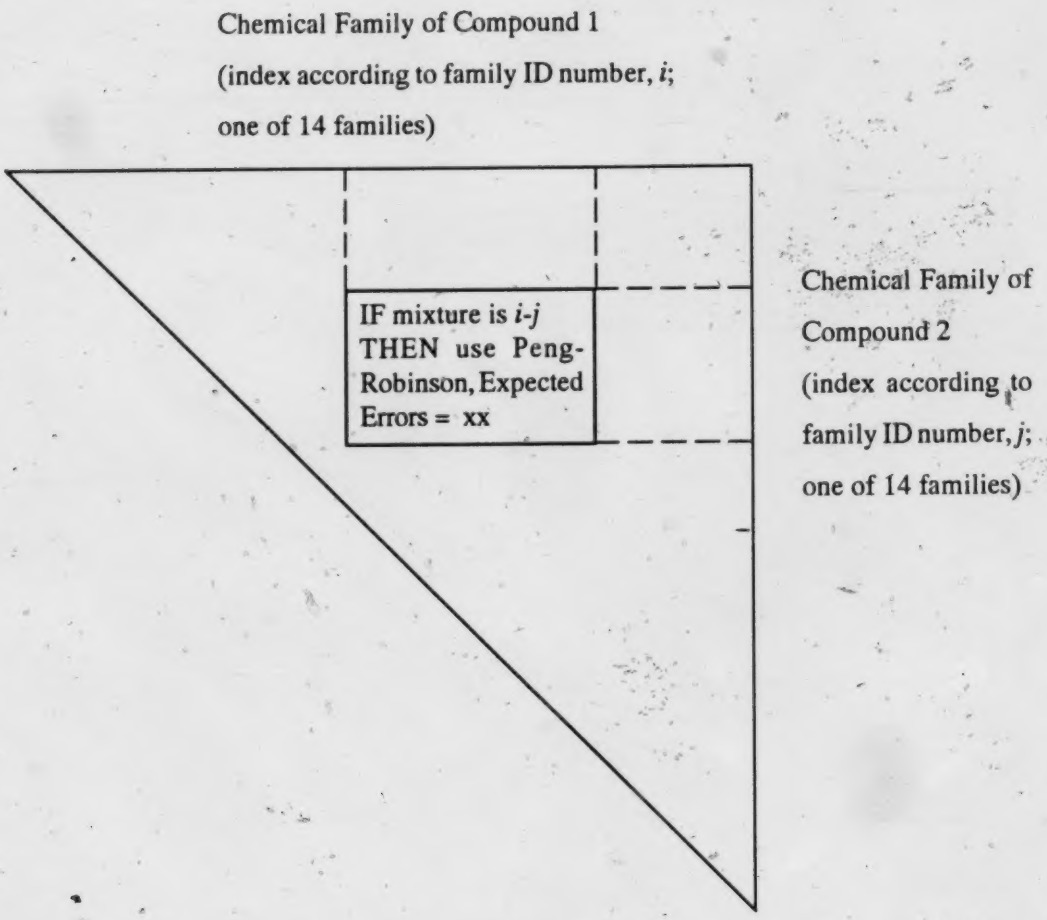

Chemical Family of Compound 1
(index according to family ID number, i;
one of 14 families)
IF mixture is i-j
THEN use Peng-Robinson, Expected Errors = xx
Chemical Family of Compound 2
(index according to family ID number, j;
one of 14 families)

Figure 3.6 Level 3 Error Matrix for Peng-Robinson Equation of State

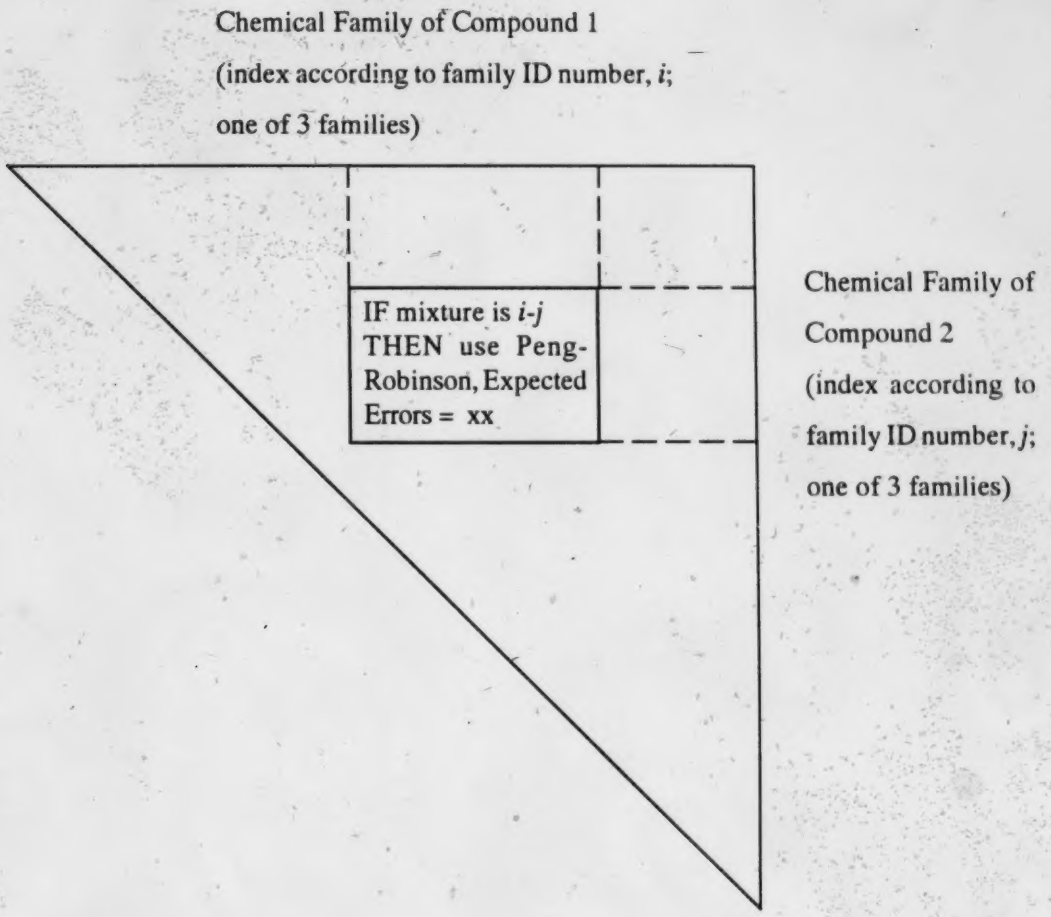

Chemical Family of Compound 1
(index according to family ID number, i;
one of 3 families)
IF mixture is i-j
THEN use Peng-Robinson, Expected Errors = xx
Chemical Family of Compound 2
(index according to family ID number, j;
one of 3 families)

Figure 3.7 Level 2 Error Matrix for Peng-Robinson Equation of State

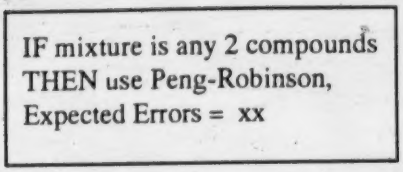


Figure 3.8 Level 1 Error Matrix for Peng-Robinson Equation of State

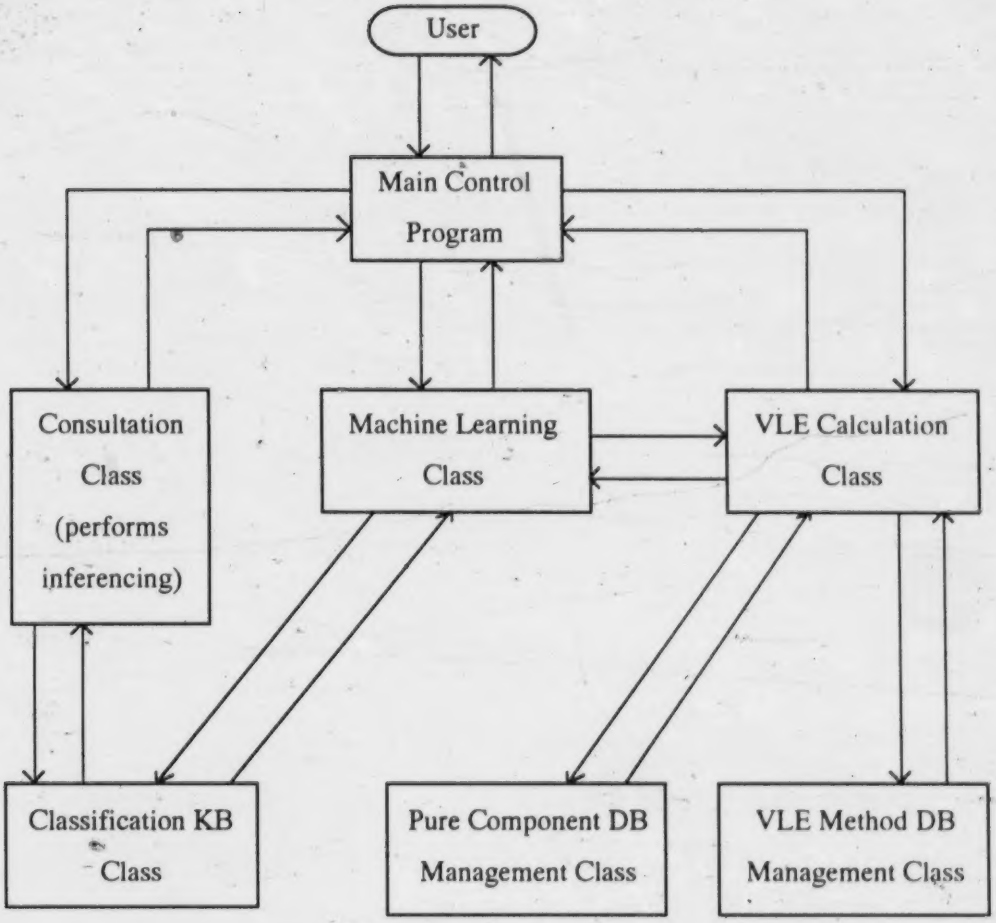


Figure 3.9 C++ Classes Built for the Expert System

is performed using training and test sets. The results of this evaluation are also presented.

3.4.1 Illustrative Examples

If no training data has been presented to the system then no recommendations can be made. When trying to consult the system, it replies with the following,

THERE IS NO KNOWLEDGE WITHIN THE KNOWLEDGE BASE.

To illustrate the learning process we shall introduce the following single data point for training.

Methanol (Compound 1)
Water (Compound 2)
x1 = 0.321 (liquid phase mol fraction of compound 1)
y1 = 0.680 (liquid phase mol fraction of compound 2)
T = 350.55 K
P = 101325 Pa
(Reference: Gmehling et al., 1982)

The calculation results are shown in Table 3.6. The Table has two listings. The first is listing summarizes the error in composition. The second summarizes the error in temperature. From the table, it can be seen that the least error in composition is by the Margules & Peng-Robinson method. The lowest temperature error is from the Wilson method.

The knowledge base now contains the information provided by this one data point. Figure 3.10 shows which cells of the error matrices now have data. The rest are still empty.

Table 3.6 Summary of Calculation Results for Example Data Point 1

Sorted by Composition Error

Phase Equilibrium Method: Margules & Peng-Robinson
Error in y: 0.037949 Error in T: 5.345428

Phase Equilibrium Method: van Laar & Peng-Robinson
Error in y: 0.037971 Error in T: 5.346130

Phase Equilibrium Method: Margules & Redlich-Kwong
Error in y: 0.038048 Error in T: 5.357391

Phase Equilibrium Method: van Laar & Redlich-Kwong
Error in y: 0.038070 Error in T: 5.358093

Phase Equilibrium Method: Margules & Soave-Redlich-Kwong
Error in y: 0.038512 Error in T: 5.426086

Phase Equilibrium Method: van Laar & Soave-Redlich-Kwong
Error in y: 0.038534 Error in T: 5.426788

Phase Equilibrium Method: Margules
Error in y: 0.039723 Error in T: 5.707825

Phase Equilibrium Method: van Laar
Error in y: 0.039745 Error in T: 5.708496

Phase Equilibrium Method: Peng-Robinson
Error in y: 0.070849 Error in T: 9.955811

Phase Equilibrium Method: Soave-Redlich-Kwong
Error in y: 0.077198 Error in T: 9.008759

Phase Equilibrium Method: Wilson
Error in y: 0.080188 Error in T: 0.892975

Phase Equilibrium Method: Wilson & Soave-Redlich-Kwong
Error in y: 0.081264 Error in T: 1.196991

Phase Equilibrium Method: Wilson & Redlich-Kwong
Error in y: 0.081699 Error in T: 1.275909

Phase Equilibrium Method: Wilson & Peng-Robinson
Error in y: 0.081781 Error in T: 1.288727

Phase Equilibrium Method: Redlich-Kwong
Error in y: 0.169481 Error in T: 52.874969

Sorted by Temperature Error

Phase Equilibrium Method: Wilson
Error in y: 0.080188 Error in T: 0.892975

Phase Equilibrium Method: Wilson & Soave-Redlich-Kwong
Error in y: 0.081264 Error in T: 1.196991

Phase Equilibrium Method: Wilson & Redlich-Kwong
Error in y: 0.081699 Error in T: 1.275909

Phase Equilibrium Method: Wilson & Peng-Robinson
Error in y: 0.081781 Error in T: 1.288727

Phase Equilibrium Method: Margules & Peng-Robinson
Error in y: 0.037949 Error in T: 5.345428

Phase Equilibrium Method: van Laar & Peng-Robinson
Error in y: 0.037971 Error in T: 5.346130

Phase Equilibrium Method: Margules & Redlich-Kwong
Error in y: 0.038048 Error in T: 5.357391

Phase Equilibrium Method: van Laar & Redlich-Kwong
Error in y: 0.038070 Error in T: 5.358093

Phase Equilibrium Method: Margules & Soave-Redlich-Kwong
Error in y: 0.038512 Error in T: 5.426086

Phase Equilibrium Method: van Laar & Soave-Redlich-Kwong
Error in y: 0.038534 Error in T: 5.426788

Phase Equilibrium Method: Margules
Error in y: 0.039723 Error in T: 5.707825

Phase Equilibrium Method: van Laar
Error in y: 0.039745 Error in T: 5.708496

Phase Equilibrium Method: Soave-Redlich-Kwong
Error in y: 0.077198 Error in T: 9.008759

Phase Equilibrium Method: Peng-Robinson
Error in y: 0.070849 Error in T: 9.955811

Phase Equilibrium Method: Redlich-Kwong
Error in y: 0.169481 Error in T: 52.874969

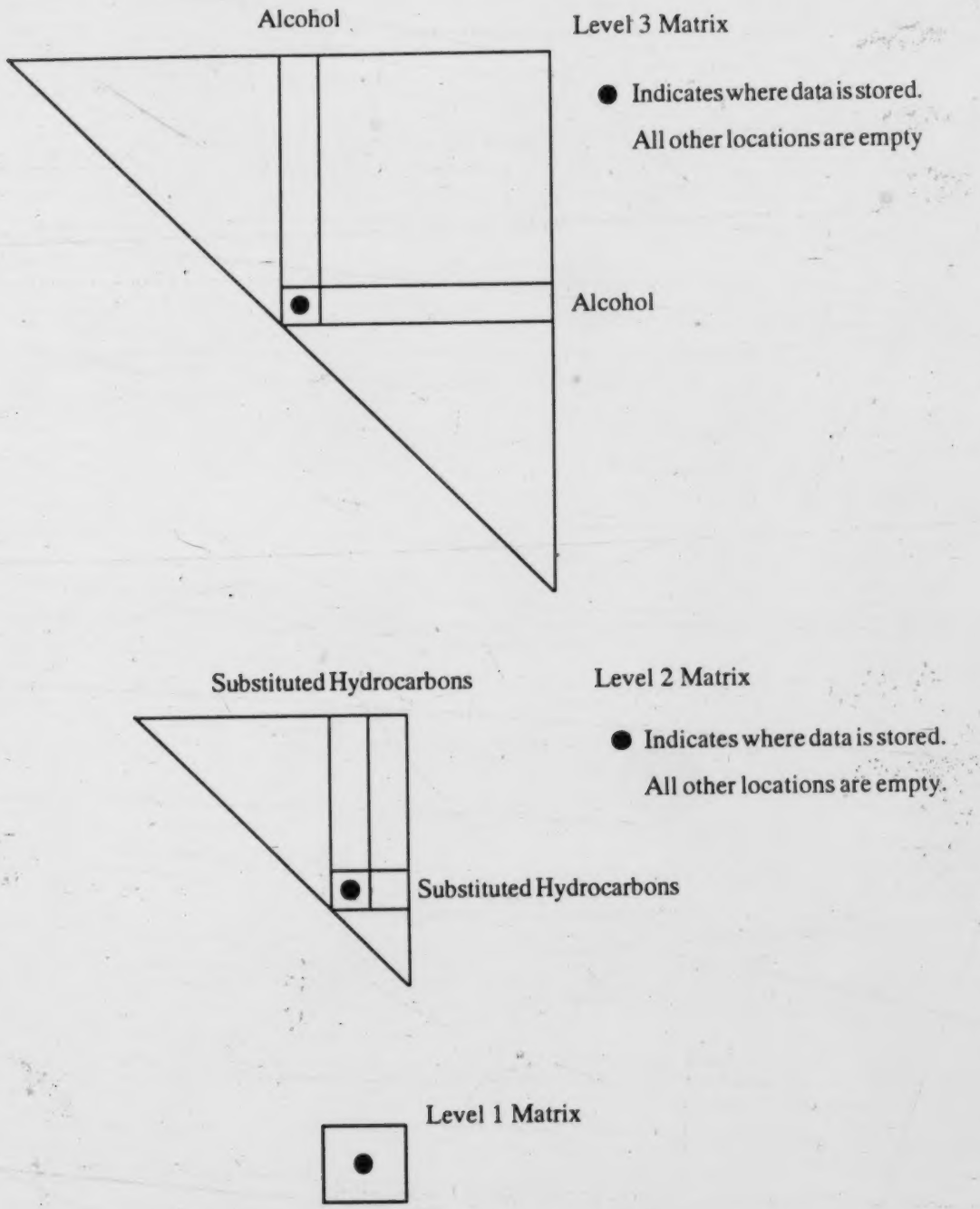


Figure 3.10 Effect of Example Data Point 1 on the Error Matrices

The following is a summary of the results from several consultations.

1) Compounds:

Methanol & Water

Recommendations:

Margules & Peng-Robinson for lowest composition error

Wilson for lowest temperature error

2) Compounds:

Butane & Pentane

Recommendations:

Margules & Peng-Robinson for lowest composition error

Wilson for lowest temperature error

3) Compounds:

Methanol & Benzene

Recommendations:

Margules & Peng-Robinson for lowest composition error

Wilson for lowest temperature error

Note that the recommendation is the same for all three pairs of compounds. The reason is that the knowledge in the system is too shallow to differentiate between types of compounds. Figures 3.11 to 3.13 show the inference process through the error matrices. Note how the inferencing moves to more general classifications when more specific data is not available.

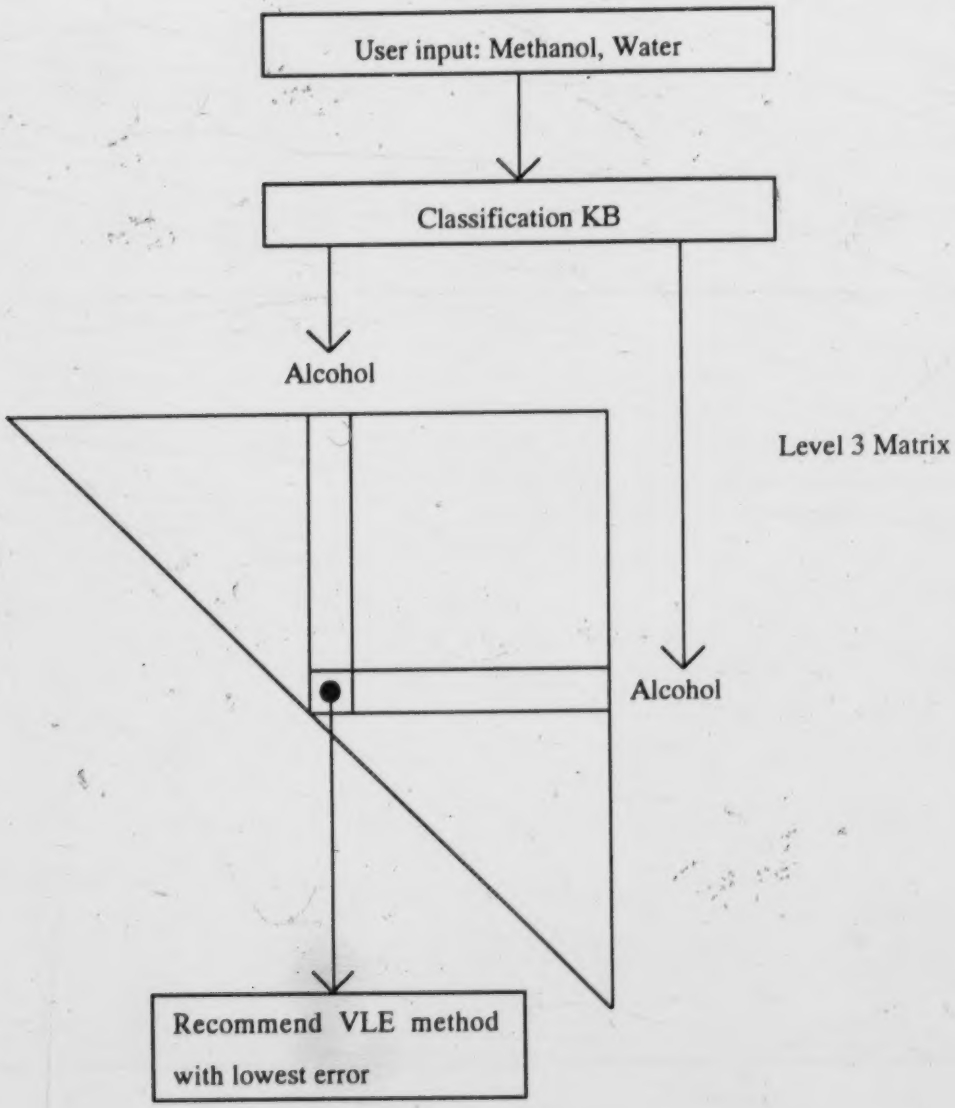


Figure 3.11 Inference Process for Methanol-Water Consultation after Training with Example Data Point 1

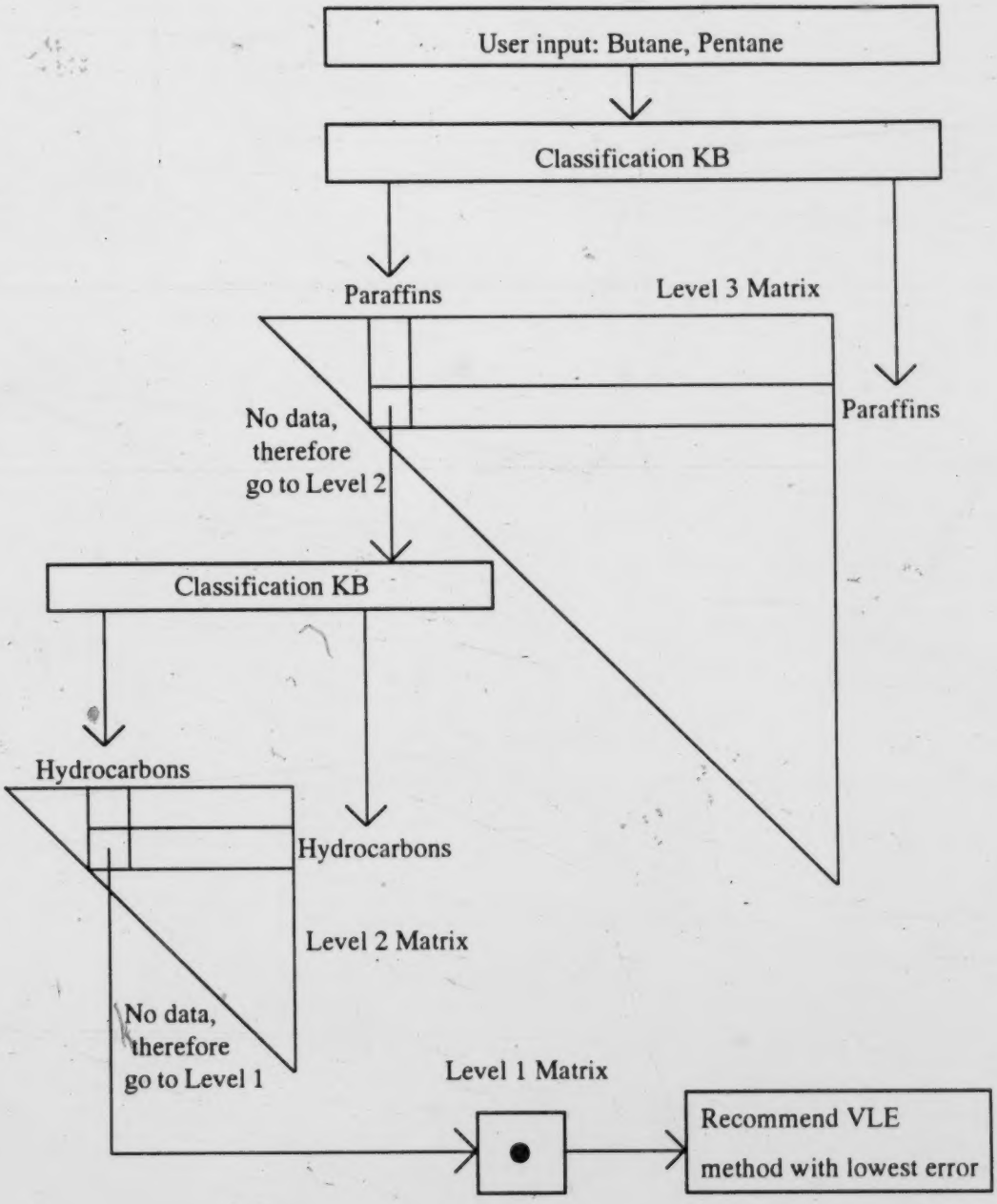


Figure 3.12 Inference Process for Butane-Pentane Consultation after Training with Example Data Point 1

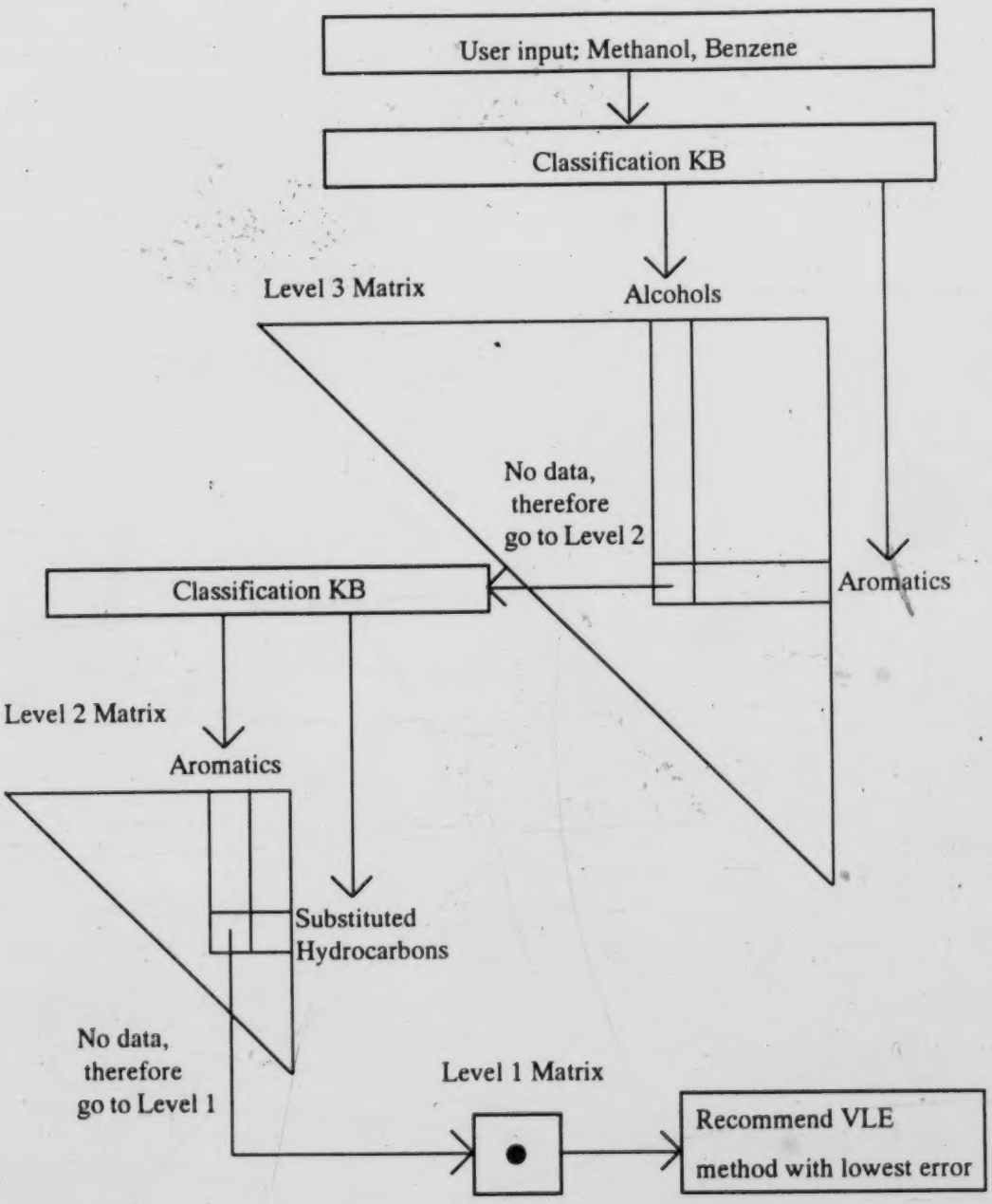


Figure 3.13 Inference Process for Methanol-Benzene Consultation after Training with Example Data Point 1

The solution is to add more training data to the system. We will add the following single data point.

Butane (Compound 1)
Pentane (Compound 2)
x1 = 0.0554
y1 = .2105
T = 298.15 K
P = 79993 Pa
(Reference: Gmehling et al., 1982)

The calculation results are shown in Table 3.7. The data has been sorted in the same manner as before. The van Laar method gives the lowest error for composition. Lowest temperature error is from the Wilson & Redlich-Kwong method. The previous recommendation for the Butane-Pentane mixture does not match with these results. Figure 3.14 shows the new state of the knowledge base. Note how results from both training points have been averaged into the Level 1 matrix.

The same set of consultations as above were made again. The increased knowledge of the system improved the recommendations. The following are the recommendations from the system.

1) Compounds:
Methanol & Water

Recommendations:
Margules & Peng-Robinson for lowest composition error
Wilson for lowest temperature error

Table 3.7 Summary of Calculation Results for Example Data Point 2

Sorted by Composition Error

Phase Equilibrium Method: van Laar
Error in y: 0.037685 Error in T: 0.724945

Phase Equilibrium Method: Margules
Error in y: 0.037694 Error in T: 0.725281

Phase Equilibrium Method: van Laar & Soave-Redlich-Kwong
Error in y: 0.038450 Error in T: 0.023621

Phase Equilibrium Method: Margules & Soave-Redlich-Kwong
Error in y: 0.038459 Error in T: 0.023315

Phase Equilibrium Method: van Laar & Peng-Robinson
Error in y: 0.038659 Error in T: 0.140808

Phase Equilibrium Method: van Laar & Redlich-Kwong
Error in y: 0.038665 Error in T: 0.107330

Phase Equilibrium Method: Margules & Peng-Robinson
Error in y: 0.038668 Error in T: 0.140472

Phase Equilibrium Method: Margules & Redlich-Kwong
Error in y: 0.038674 Error in T: 0.106964

Phase Equilibrium Method: Soave-Redlich-Kwong
Error in y: 0.044564 Error in T: 1.026550

Phase Equilibrium Method: Peng-Robinson
Error in y: 0.046340 Error in T: 0.805908

Phase Equilibrium Method: Wilson
Error in y: 0.050424 Error in T: 0.836212

Phase Equilibrium Method: Wilson & Soave-Redlich-Kwong
Error in y: 0.051205 Error in T: 0.084991

Phase Equilibrium Method: Wilson & Peng-Robinson
Error in y: 0.051414 Error in T: 0.028717

Phase Equilibrium Method: Wilson & Redlich-Kwong
Error in y: 0.051415 Error in T: 0.002441

Phase Equilibrium Method: Redlich-Kwong
Error in y: 0.068418 Error in T: 15.035736

Sorted by Temperature Error

Phase Equilibrium Method: Wilson & Redlich-Kwong
Error in y: 0.051415 Error in T: 0.002441

Phase Equilibrium Method: Margules & Soave-Redlich-Kwong
Error in y: 0.038459 Error in T: 0.023315

Phase Equilibrium Method: van Laar & Soave-Redlich-Kwong
Error in y: 0.038450 Error in T: 0.023621

Phase Equilibrium Method: Wilson & Peng-Robinson
Error in y: 0.051414 Error in T: 0.028717

Phase Equilibrium Method: Wilson & Soave-Redlich-Kwong
Error in y: 0.051205 Error in T: 0.084991

Phase Equilibrium Method: Margules & Redlich-Kwong
Error in y: 0.038674 Error in T: 0.106964

Phase Equilibrium Method: van Laar & Redlich-Kwong
Error in y: 0.038665 Error in T: 0.107330

Phase Equilibrium Method: Margules & Peng-Robinson
Error in y: 0.038668 Error in T: 0.140472

Phase Equilibrium Method: van Laar & Peng-Robinson
Error in y: 0.038659 Error in T: 0.140808

Phase Equilibrium Method: van Laar
Error in y: 0.037685 Error in T: 0.724945

Phase Equilibrium Method: Margules
Error in y: 0.037694 Error in T: 0.725281

Phase Equilibrium Method: Peng-Robinson
Error in y: 0.046340 Error in T: 0.805908

Phase Equilibrium Method: Wilson
Error in y: 0.050424 Error in T: 0.836212

Phase Equilibrium Method: Soave-Redlich-Kwong
Error in y: 0.044564 Error in T: 1.026550

Phase Equilibrium Method: Redlich-Kwong
Error in y: 0.068418 Error in T: 15.035736

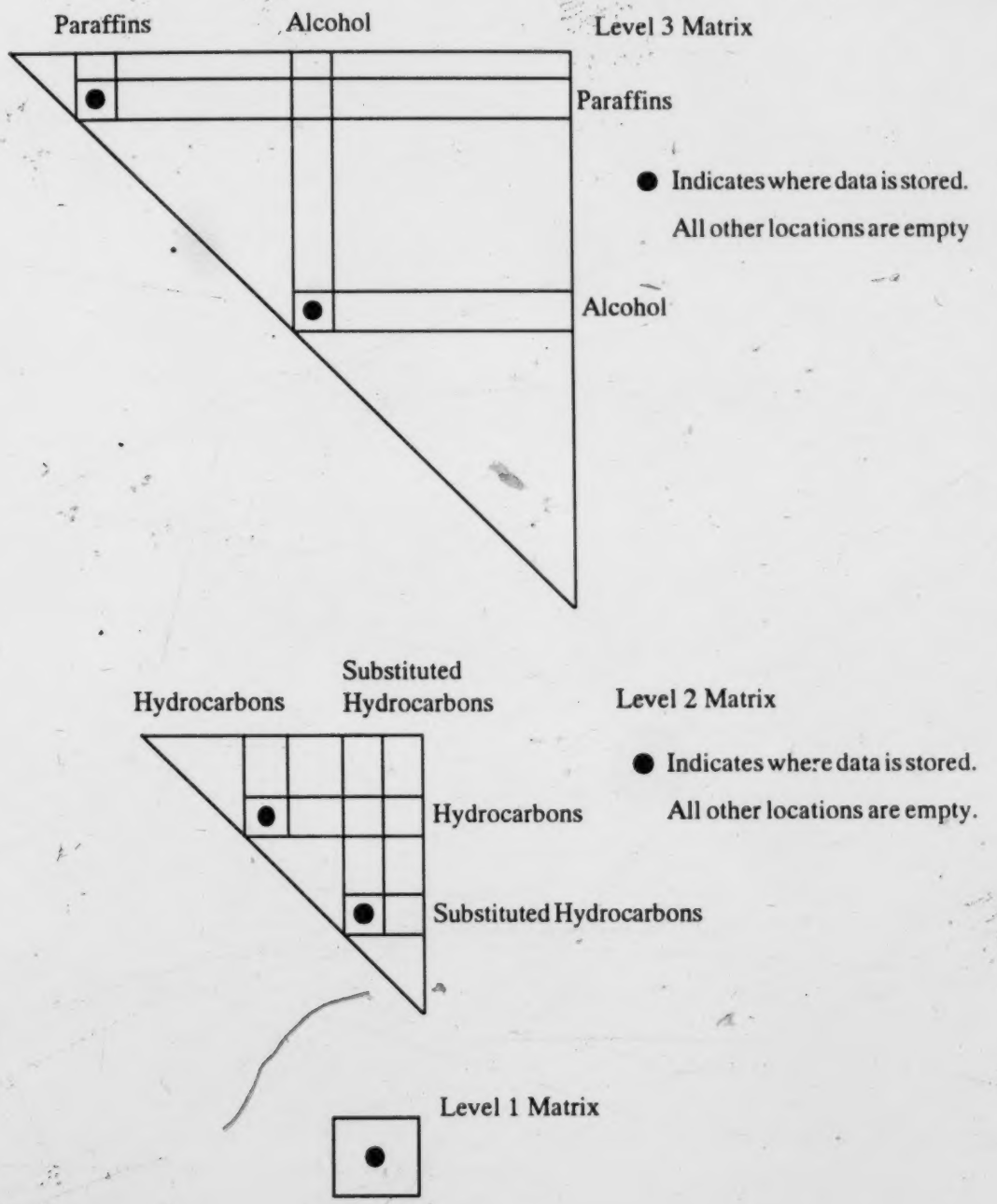


Figure 3.14 New State of the Error Matrices after Training with Example Data Point 2

2) Compounds:

Butane & Pentane

Recommendations:

Van Laar for lowest composition error

Wilson & Redlich-Kwong for lowest temperature error

3) Compounds:

Methanol & Benzene

Recommendations:

Wilson & Redlich-Kwong for lowest composition error

Wilson & Soave-Redlich-Kwong for lowest temperature error

The recommendations for all three methods are now different. Figures 3.15 to 3.17 show the inference process for each of these consultations. The recommendation for case 3 is based on the average from both training points. Since no data specific to case 3 has been presented, the system uses the results from other data to make a recommendation.

3.4.2 Evaluation of the Expert System

Evaluation of expert system performance is a difficult task. It is hard to evaluate the results from an expert system because the information is symbolic in nature. A test was developed to evaluate the performance of the expert system developed here. The objectives of the test was to evaluate the success of the learning algorithm (Kibler and Langley, 1988). This test does not perform an exhaustive evaluation of the expert system, but it does provide

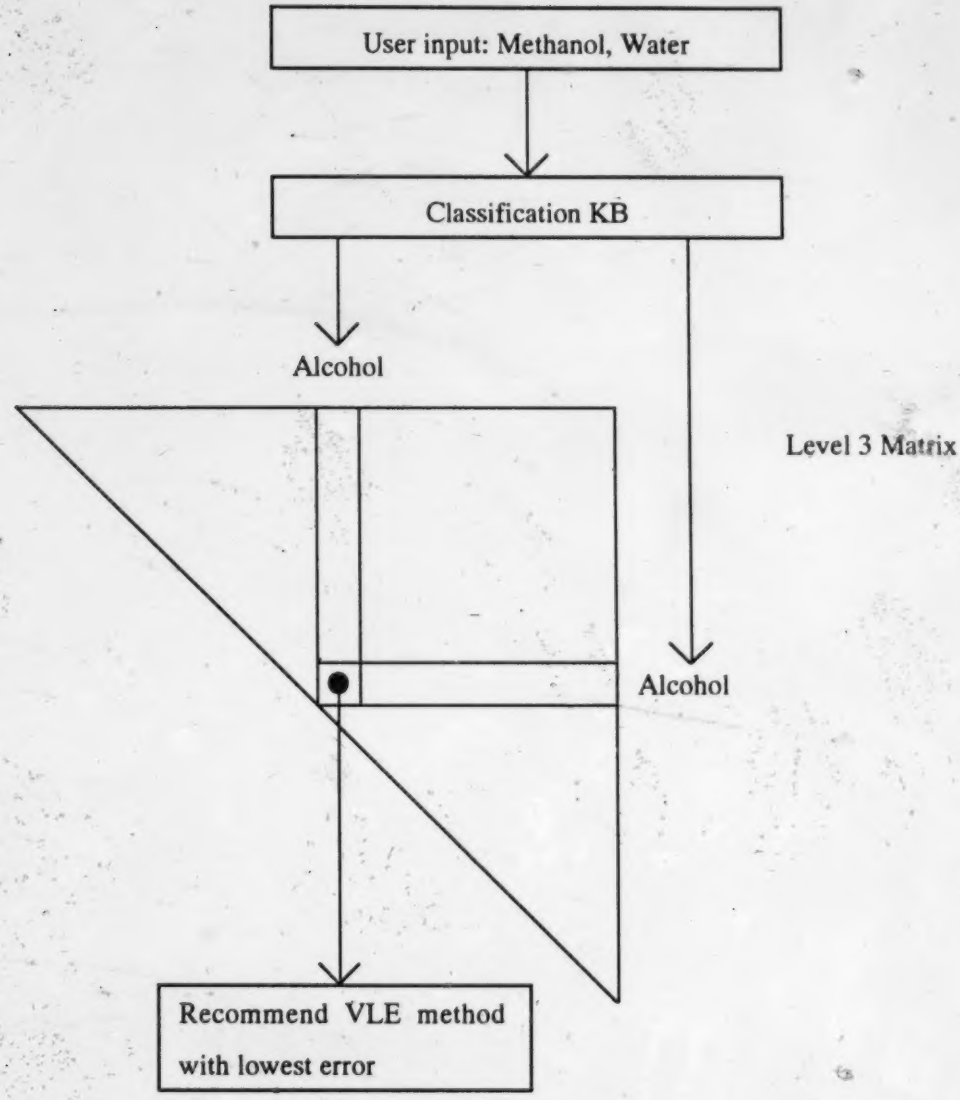


Figure 3.15 Inference Process for Methanol-Water Consultation after Training with Example Data Point 2

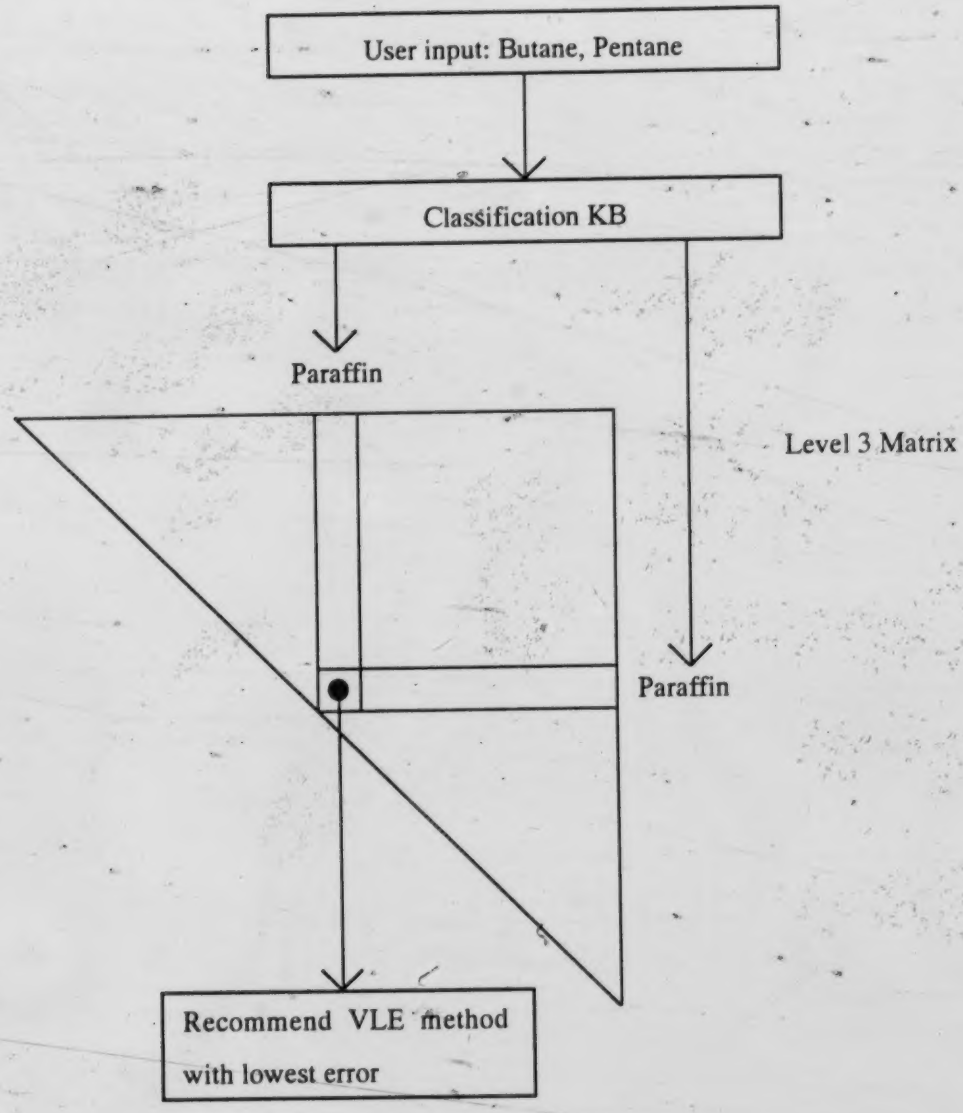


Figure 3.16 Inference Process for Butane-Pentane Consultation after Training with Example Data Point 2

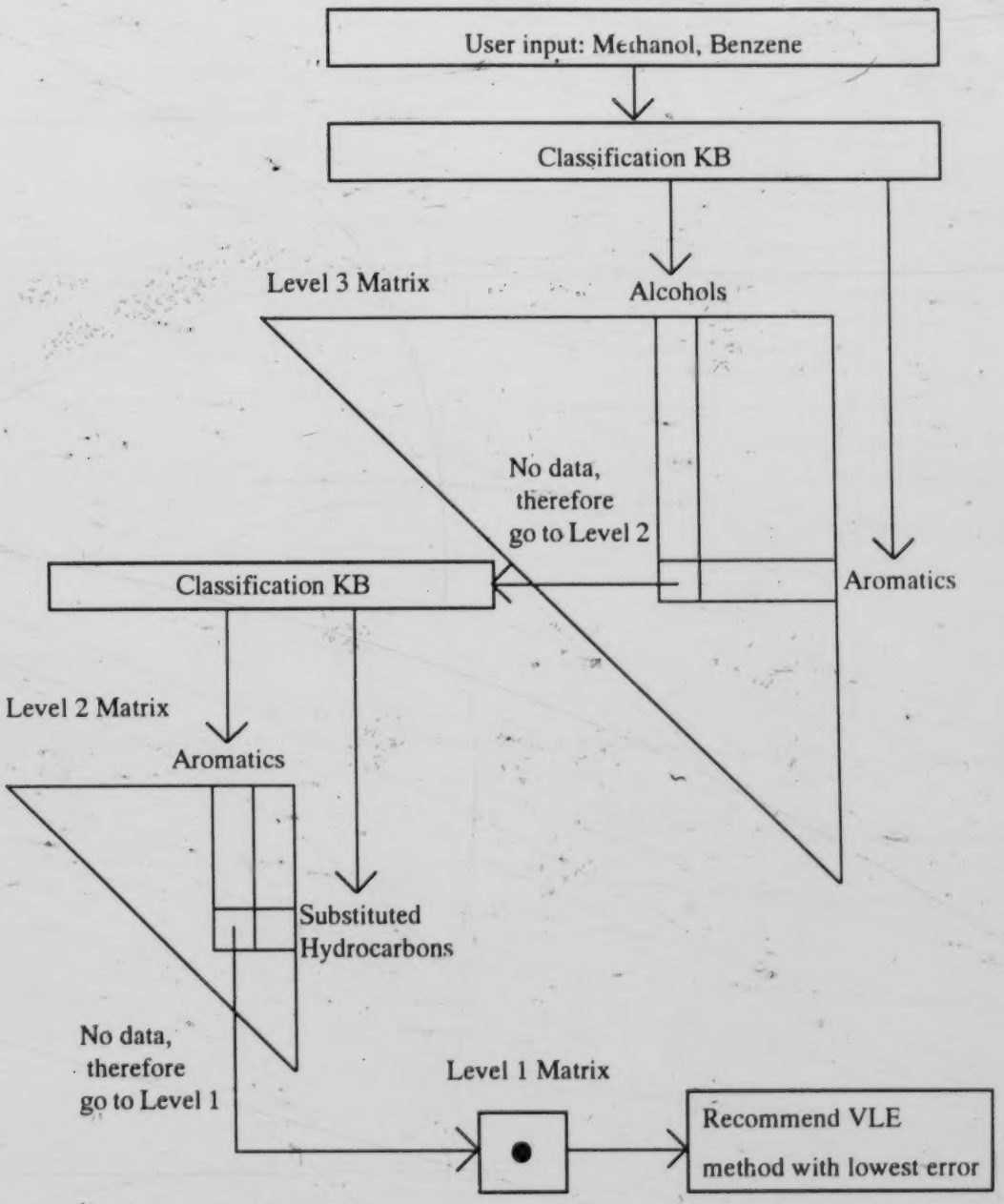


Figure 3.17 Inference Process for Methanol-Benzene Consultation after Training with Example Data Point 2

an evaluation of the algorithms used.

The test used data from a testbed of 396 data points. The data points represent phase envelopes for 26 different binary mixtures. They were obtained from the DECHEMA Vapor-Liquid Equilibrium Library (Gmehling et al., 1982). Most of the data contained in the library were tested using two different thermodynamic consistency tests. Only data that passed both tests were included in this testbed.

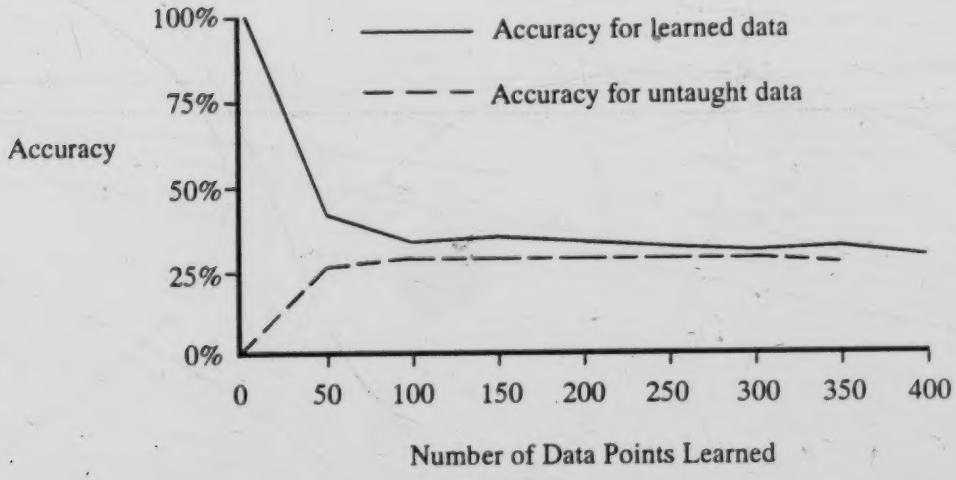
The testbed was divided evenly into 8 sets. Each set had a cross-section of data points from each of the 26 binary mixtures. The 8 sets were then supplied to the learning algorithm in random order. After each set was supplied, the expert system was requested to provide VLEC method advice for all the test sets. In this manner, the system could be tested for accuracy. The recommendations were compared with calculation results to determine if they matched with the VLEC method that provided the least error. Each test provided three results. One was the accuracy for data that had been presented to the learning algorithm. The second was accuracy for data similar to previously learned examples. The third result was the change in accuracy as the amount of learned data increased.

Table 3.8 and Figure 3.18 show the average results from 8 tests. The first point to notice is that a steady prediction accuracy is achieved after about 100 training points. The steady-state accuracies were about 30% and 54% for composition and temperature predictions, respectively. The solid lines show the prediction accuracies for the data which had been learned. They begin at 100% for a single data point and decay as more data is presented. Dotted lines show the accuracies for data points which had not been taught to the system. This data was for the same chemical compounds as data previously taught to the system. They begin at 0% when there is no knowledge in the system and increase in accuracy as the system expands its knowledge base. These results imply that after about 100 training points, consistent prediction accuracies are obtained. The consistent accuracies are for previously

Table 3.8 Average Number of 100% Correct Recommendations During 8 Test Sessions

Number of Learned Data Points	Composition Predictions		Temperature Predictions	
	Learned Data	Data Not Learned	Learned Data	Data Not Learned
50	42%	26%	56%	46%
100	33%	28%	50%	44%
150	35%	29%	55%	50%
200	33%	29%	55%	50%
250	33%	28%	53%	49%
300	31%	29%	54%	50%
350	31%	27%	54%	51%
400	30%	N/A	54%	N/A

a. Accuracy for Composition Recommendations



b. Accuracy for Temperature Recommendations

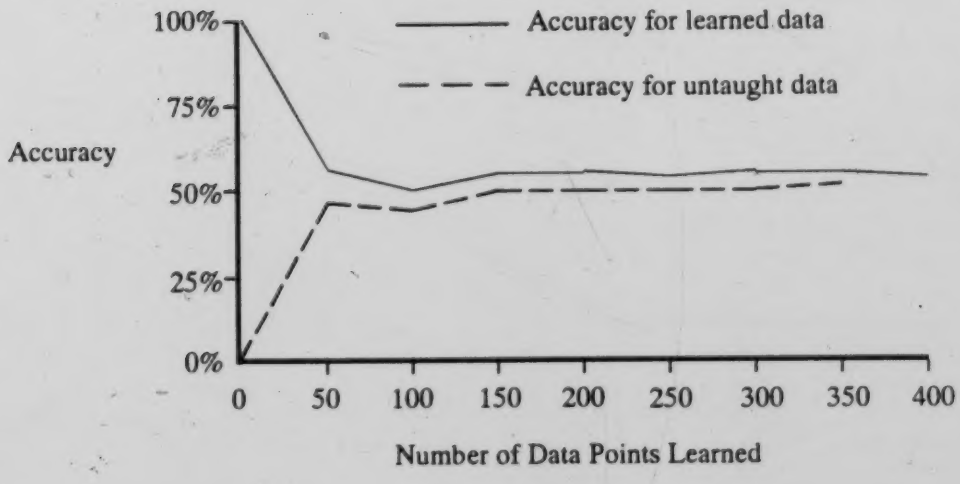


Figure 3.18 Average Number of 100% Correct Accuracies During 8 Test Sessions

learned data and data which is related to previously learned data. This expert system can learn from training data, then provide recommendations on the related, but not learned, data with similar accuracies.

In all the test cases the prediction accuracy for temperature recommendations was higher than the composition recommendations. This indicates that one VLEC method is consistently more accurate for temperature predictions for a given pair of chemical families. The corollary of this is that one VLEC method does not provide the best accuracy for the entire range of compositions. The training data consisted of phase envelope data. At either constant pressure or temperature the liquid compositions are varied and the vapor composition is measured. The compositions range from dilute to concentrated. The higher temperature accuracy indicates that one VLEC method is more consistent in providing the most accurate calculations. On the other hand, composition accuracy is dependent on concentration. One VLEC method provides the best accuracy over smaller portions of the composition range. This may indicate that a composition classification should be added to the chemical family classifications. Another explanation for this result is provided from Tables 3.6 and 3.7. Both tables show that the first 8 VLEC methods sorted for composition errors all provide very similar results. This large degree of competition reduces the dominance of any one VLEC method. The results sorted by temperature errors show that the first 4 methods have similar results. The reduced competition level improves the selection of one dominant VLEC method. Although these tables only represent two calculations, they are representative of the entire test set.

The level of accuracies do not reflect the confidence of the expert system recommendations. These tests evaluate the expert system's ability to recommend the most accurate method. They do not evaluate how often the expert system makes satisfactory recommendations. The recommendations from the expert system may not always be the most accurate for a given situation but they may still provide sufficiently accurate calculation results. The

trends shown in Figures 3.6 and 3.7, where several methods provide similar results, indicate that several methods may be capable of providing accurate calculation results. There are two possibilities for evaluating the quality of the expert system. The first is to set some tolerable level of calculation error and test how often the expert system recommends a VLEC method that provides this level of accuracy. The second is to request a human expert to evaluate the system. However, it should be noted that the recommendation accuracies are both significantly above the random chance level. The random chance accuracy would be 6.7% (1 in 15). Since the system performs much better, it represents a successful example of machine learning.

3.5 Discussion

The applications of this system are numerous. The range of compounds in the database make this system flexible. It could provide advice during the design or simulation of flash drums. Distillation is another area requiring good VLE prediction. This system could be used for simulations or the early stages of design. The VLEC method selection for low error in composition prediction could be used to identify theoretical stage requirements. This would give a starting point for a design. During simulations, this would determine if a column could perform its task under a given set of conditions. Distillation columns are very expensive and a tighter design could save thousands of dollars. The low temperature error recommendation could be used to determine the utility requirements for a given column. Two phase flow also requires prediction of VLE data.

The architecture developed for this system was compact and efficient for both the learning and the consultation functions. The benefits of this system's architecture may be applied in other areas. The design could be used for any domain that has many predictive methods available. Figure 3.19 shows the general architecture of this system. Other areas

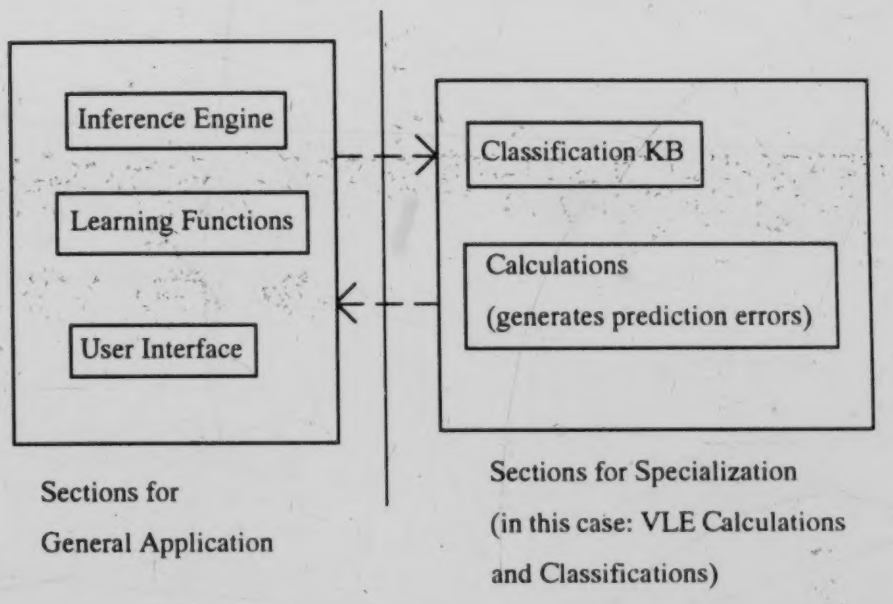


Figure 3.19 General Architecture of this Expert System

where this system could be implemented:

1) viscosity
2) vapor pressure
3) liquid density
4) heat capacity
5) thermal conductivity
6) diffusion coefficients
7) surface tension
8) distillation simulation
9) flash calculations

The calculation section in Figure 3.19 would be replaced by the appropriate predictive methods for the given application. The classification rules and the error matrices may also require modification.

The error matrices represent the confidence factors for the recommendation rules. In this system the confidence factors have been directly related to a property of the domain. Most implementations of confidence factors have only a subjective relationship to domain properties. The difference here was the knowledge representation. The knowledge structure was designed so confidence factors could be related to the prediction errors of the VLEC methods.

Relating the confidence factors and the calculation errors produced a high degree of integration of symbolic and numeric processing. The symbolic information of the knowledge base required the results of the calculations to add credence to the knowledge. The numeric calculations required the symbolic knowledge structure to interpret and classify the calculation results. Since engineering involves both numeric and symbolic processing, this system

characterizes the future trends in engineering expert systems. Another recent example of this trend is provided by Belkin and Stengel (1990). Their object was to implement machine learning to take numeric data and derive symbolic rules for aircraft navigation assistance. The techniques employed were different than those implemented in this system but the overall approach and objective are similar.

There is a requirement for a good classification system for chemical compounds. The classification system may be different for various applications. For VLE prediction the classification system should be sensitive to the VLEC methods and the type and source of errors. Errors in family classification occur when a compound contains features which classify it into one family but its behavior is dominated by other features. An example is n-decanol. This molecule contains an alcoholic OH group. A simple classification system would classify it as an alcohol. However, the rest of the molecular chain is large enough to reduce the effect of the hydrogen bonding. The net result is a behavior which is between a non-alcohol and alcohol. Another error could arise when a compound contains several active and distinguishing groups. The task is to identify the dominant feature and to classify the compound accordingly. This type of problem is suited to fuzzy logic. Fuzzy logic would allow chemicals to belong to several families. The error calculations and the recommendations would have to be modified to handle the fuzzy logic.

The chemical family classification system may be modified after the system has been developed. If some compounds have changed their classification, then the error history of the VLEC methods is no longer completely accurate. There are two approaches to correcting this situation. The first is to scrap the knowledge base and retrain the system with the data in the training set database. This may be time consuming if the system has a lot of training data. The alternative is to ignore the change in classifications. The philosophy behind this approach is that the system will eventually average out the old data which was based on compounds no longer contained within the current classification. This could be viewed as

forgetting old data. This second approach is more feasible. To accelerate the forgetting process we could add extra weighting to the new training examples.

With machine learning there is the possibility of noise in the training set. Noise in this context is inaccurate experimental data (Quinlan, 1986). Inaccurate data will propagate into the confidence levels of the VLEC methods and influence any recommendations made by the expert system. Noise may arise from several areas, such as inaccurate experimental data or inaccurate sources of data. If a reliable source is found for experimental data, this noise can be effectively ignored. There are several reliable sources of VLE data and these databases are used internationally. Two popular ones are the DECHEMA and DIPPR databases. They thoroughly test and investigate all the data before it becomes available to the user. There is still a chance of noise but small levels will be averaged out over time.

3.6 Conclusions

An expert system for the selection of vapor-liquid equilibrium (VLE) prediction methods has been successfully developed. It acquires knowledge through the technique of machine learning. Training sets of experimental data are presented to the system. It then predicts the data using the VLE calculation methods in its database. The prediction errors form the basis of the knowledge used for recommendations.

Through testing it was found that the expert system provides different accuracies for composition and temperature recommendations. Composition recommendations are 100% correct about 30% of the time. Temperature recommendations are 100% correct about 50% of the time. The superior performance for the temperature recommendations were caused by a wider error range for the VLE calculation methods when predicting temperature. Fewer VLE calculation methods provided small errors so a superior method could be more easily

identified. After about 100 training points the expert system provided similar accuracies for learned data and data which had not been learned. Supplying more than 100 training points did not improve or degrade the system's accuracies.

This system extends the previous work in this area in two distinct ways. The first is the implementation of machine learning. The second is the integration of symbolic and numeric processing. Machine learning is used to modify symbolic rules from numerical prediction errors. The interdependence between numeric and symbolic information processing indicates a high degree of integration. This type of integration is a desired characteristic for engineering-based expert systems.

The system architecture is flexible. Any domain involved with prediction and correlation of real data could use this architecture for an expert system. An inference mechanism specifically catered to this architecture has been developed as well.

The knowledge representation was designed to be compact and efficient. Its main benefit was that it allowed the knowledge base to be directly related to a domain property. In this case, the domain property was the predictive errors of the various VLE methods.

It was demonstrated that the degree of generalization in the classification methods can affect the deductive ability of an expert system. More general classifications result in wider application of the knowledge. However, reliability may be lost with more general classifications. The more detailed the classifications, the closer to reality the knowledge becomes.

4. Case Study Two: An Expert Advisory System for Safe Plant Startup*

This chapter describes the second case study in applied artificial intelligence in chemical engineering. An expert advisory system was developed to provide advice during the startup of a section of a refinery.

4.1 Introduction

Most expert system research in chemical engineering has been for process control or design (Stephanopoulos and Davis, 1990). Process operation support is another area where expert systems could be of value. Capturing the knowledge of experienced operations experts is very challenging because of the complicated procedures needed and the potential hazards to the operators. An expert system prototype was developed through a cooperative research effort of the University of Alberta's Intelligence Engineering Laboratory and an industrial partner.

The objective of this research was to provide an expert system capable of recommending the proper and safest startup procedures. Many injuries and fatalities in the chemical process industries have been due to human errors during the operation of a plant (Kletz, 1988). It has been emphasized that proper procedures must be followed during plant operation (Croce et al., 1988). Following proper procedures reduces the chance of incidents. With the use of computer control for plants it was hoped that increased safety would result. Some increase in safety is achieved but other hazards arise. This is due to the computer's inability

---

* Two versions of this chapter have been accepted at two conferences. Ref:

1) **Dong, R.**, M. Rao, H. Bacon and V. Mahalec. (1991). "An Expert System for Refinery Plant Operation." Proc. of the IFAC Symposium on Distributed Intelligence Systems, August 13-15 1991, Arlington, Virginia, USA.

2) **Dong, R.**, M. Rao, H. Bacon and V. Mahalec. (1991). "An Expert Advisory System for Safe Plant Startup." Fourth International Symposium on Process Systems Engineering, August 5-9, 1991, Montebello, Quebec, Canada.

to perform as a human operator. It lacks the intelligence and expertise of the operation experts (Pearson and Brazendale, 1988). This work concentrates on the development of an intelligent system to provide advice on the selection of proper and safe procedures given the current state of the plant.

This prototype was developed as an initial stage in the overall development of an online, real time expert system for refinery operation. The project plan is summarized as follows:

1) develop an offline advisory system prototype

2) develop an offline, full featured advisory system

3) develop an offline advisory system that accepts and retrieves real time data from the process

4) develop an offline advisory system that provides real time operation support

5) develop an online, real time expert system for plant operation.

By following this development schedule the design and integrity of an online, real time expert system will be more robust. The multiple stages ensure thorough testing of the knowledge base and improve acceptance by the process operators.

4.2 Background

Very little previous research has been done in the area of automatic plant startup. A literature search did not produce any references. This indicates that this is original research work. There has been related research in the fault diagnosis area. These researchers concentrated on identifying faults during the normal operation of a plant (Kramer, 1987; Talukdar, Cardozo and Leao, 1986; Abu el Ata-Doss and Brunet, 1986). The research

presented in this chapter concentrates on the correct selection of procedures during the startup of a plant.

4.2.1 Process Description

The unit under consideration is a gas oil treater used to produce a low pour oil. The two main process streams contain oil and hydrogen, each under high pressure and temperature. These streams, as well as the fuel and high pressure steam lines, pose the greatest potential for danger in operating the unit. A simplified schematic of the process is shown in Figure 4.1.

The feed oil comes from another area of the plant and is passed through a preheat train. This consists of a series of heat exchangers and one fuel gas furnace. Feed hydrogen originates from a steam reformer located at the plant. The feed hydrogen is combined with recycle hydrogen and passed through a compressor. The combined stream is heated in another fuel gas furnace and then combined with the heated feed oil. The catalytic reactor breaks the large oil molecules as well as removing the sulphur as hydrogen sulphide. The outlet from the reactor is cooled and is passed to a separator train. The first separator passes the condensate to the purification section. The overheads are cooled further and passed to another separator. The bottoms of the second separator are also sent to the purification section. The overheads are stripped of hydrogen sulphide and then returned to compression as recycle hydrogen. The initial portion of the purification stage is steam stripping. The light ends are driven out of the stripper and are sent to a debutanizer. The bottoms of the steam stripper are dehydrated in a vacuum dryer. The dried bottoms are sent to tankage as the main low pour product. The wet overheads are returned to the steam stripper as recycle oil.

There are two modes of operation for this unit. During the winter the unit operates

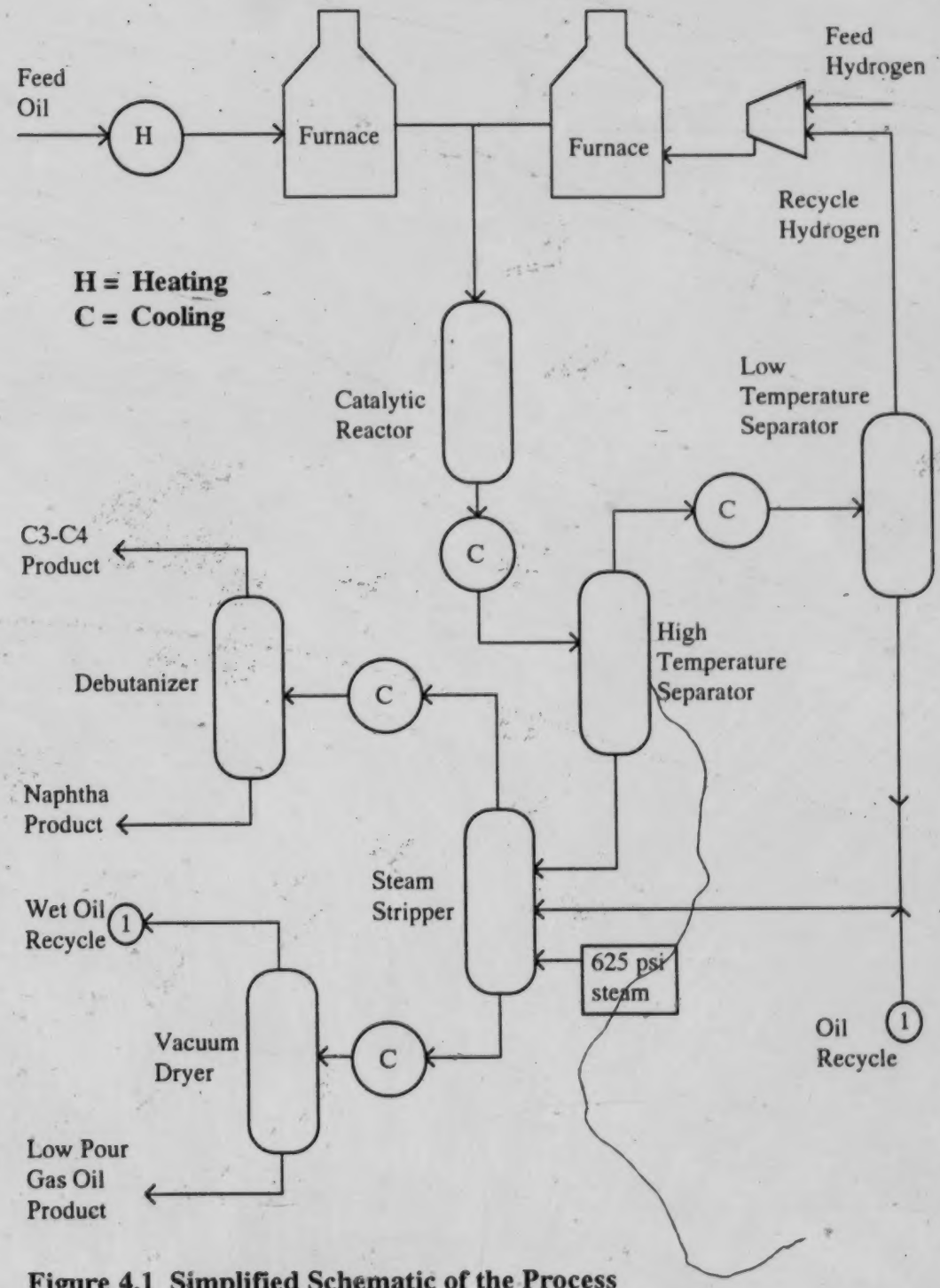


Figure 4.1 Simplified Schematic of the Process

in the manner described above. During the summer, a slightly different product is produced. The debutanizer is mothballed and not used. These two different modes of operation complicate operating the unit. These modes will be referred to as winter and summer operation.

4.2.2 Startup Procedures

Safe startups require correct procedures and good leadership from the senior operators. This unit was selected for the prototype development because it had a good set of written procedures as well as several on-site experts. The operators also accepted the investigation of new technology that would improve safety. A further criteria was that the unit has a scheduled shutdown twice a year. This regular shutdown and startup schedule ensured continuous use of a startup advisory system.

The startup procedures are made up of seven main phases. In each of these startup phases there is a set of procedures to be done. For each of the procedures there is a step-by-step set of instructions. Table 4.1 summarizes the startup phases and their procedures. In total, there are 40 procedures grouped into 7 phases. The procedures that are performed depend on the mode of operation and the type of shutdown. Shutdowns can be either total or partial. Total shutdowns are for major maintenance and to switch modes of operation. Two total shutdowns are scheduled each year. Partial shutdowns occur when minor repairs are needed. Minor repairs include valve replacements, repairing leaks and cleaning heat exchangers.

These startup phases do not include the base level utilities that are commissioned during an initial startup and left online continuously. These extra phases are very uncommon during startups other than the initial startup for a new unit. Utilities that are included in these

Table 4.1 Summary of the Startup Procedures

1) General Preparations
    a) Check Fire Extinguishers
    b) Check Safety Equipment
2) Oil Circuit Preparation, Air Free and Tightness Test (AF&TT) the following:
    a) Oil Feed Circuit
    b) Steam Stripper & Overheads
    c) Stripper Bottoms
    d) Debutanizer
    e) Vacuum Dryer
    f) Vacuum Overhead
    g) Vacuum Bottoms
    h) Recycle Oil Separator
    i) Purge Gas Knock-Out Drum
    j) Fuel Gas Circuit
3) Pre-Startup Checklists
    a) Fire Hydrants
    b) Furnace Dryout
    c) Hydrogen Circuits AF&TT
    d) Check Blanks
    e) Car Seals
    f) Safety Valves
    g) Emergency Valves
    h) Alarm Plugs
    i) Shutdown Trip Switches
    j) Analyzers
4) Establish Recirculation
    a) Back in Naphtha into Debutanizer
    b) Back in Oil into Separators
    c) Oil Fill the Feed Preheat Circuit
    d) Pressure Test Preheat with Oil
    e) Commission Oil and Vacuum Circuits
5) Catalyst Preparation
    a) Caustic Treating
    b) Presulphiding
6) Hydrogen Circuit Preparation
    a) Start Compressors
    b) Warm Up the Hydrogen Circuit with Nitrogen
    c) Pressure Test with Nitrogen
    d) Flush with Hydrogen
    e) Establish Hydrogen Recycle
7) Initiate Reacting and Slowly Ramp to Full Capacity.
    a) Warm both the Oil and the Hydrogen Circuits
    b) Commission Lean MEA
    c) Pump Feed Oil into Unit
    d) Swing Feed into Reactor
    e) Establish Debutanizer Reflux
    f) Slowly Ramp to Full Capacity

are the closed drain header, 125 psi steam header, instrument air, etc.

4.3 System Structure

The developed expert advisory system is described in this section. Certain details have not been revealed to protect the proprietary information of the partner company.

4.3.1 Chemical/Mechanical State

Before the startup procedures can be recommended, the state of the unit must be identified. The state determines how the startup procedures are affected. One of the benefits of expert system construction is that the domain knowledge becomes more organized and understood. During the knowledge base organization for this project a new model for the process resulted. The model was used to describe the state of the physical plant using three concepts: network connection, chemical state and mechanical state.

The chemical state of a unit describes chemical processing characteristics like flowrates, temperatures, pressures, types of chemicals, degree of reaction, etc. The chemical state is the parameter that must be shifted from its current state to the goal state: the normal operation for which the unit was designed. In the expert system developed here, the chemical state was used to identify the conditions for the main process streams and operations. For example, the hydrogen stream needs to be above a certain temperature and pressure before it is mixed with the oil stream and sent to the reactor. Similarly, the oil stream must also be above a certain threshold temperature and pressure. These conditions must be met before chemical reacting occurs otherwise the reactor catalyst may get poisoned.

The mechanical state describes the physical location and positioning of the equipment as well as their condition. This would be valve settings, state of blanks (in or out), vessel temperatures/pressures, location of bleed valves, elbows and other constrictions, etc. Usually, process simulation relies entirely on the chemical state of the unit. In modelling the startup problem, this is not sufficient. The startup procedures operate directly on the mechanical state to achieve changes in the chemical state. To drive the hydrogen and oil streams to their threshold conditions, bypasses and recycle routes must be setup. This is done by closing valves on the normal piping routes and opening others which redirect the hydrogen and oil away from the mixing point and reactor. Meanwhile, steam and fuel gas must be supplied to provide heat to these streams. Heat exchangers and furnaces must be operating to provide the necessary heat. Compressors and pumps must be working to add pressure to the streams. These actions represent modifications to the mechanical state and indirectly affect the chemical state.

Network connection is used to present the chemical and mechanical connections between all of the equipment. The connection description provides a means of determining the interdependencies between the equipment. The network connection concept was implemented by grouping individual equipment into circuits. These circuits represent sub-units of the plant section. Each circuit performed one specific task, such as pre-heating the feed oil. These circuits are described in more detail in section 4.3.3.

4.3.2 Knowledge Acquisition

Knowledge acquisition is the most difficult task when developing an expert system. Acquisition of the knowledge for this expert system was facilitated by several factors. First, there was an extensive set of written procedures available for the unit. These manuals were written by a member of the original design team for this plant section. The procedures had

been modified and refined over 6 years of implementation. These formed the basis for the knowledge of this expert system.

There were also several experts available for consultation. The main expert was a senior operator with many years of experience. He was part of the original design team and was the most knowledge person about this section of the refinery. As the original author of the startup procedures manual, and as the most experienced operator for this plant section, his input to the expert system was fundamental. Furthermore, this expert was nearing retirement so it was paramount to capture and retain some of his knowledge and keep it available for use within the plant.

Finally, as the main developer for the system, I am trained in chemical engineering and AI fundamentals. Through the written procedures, expert consultation and first hand experience, we were able to acquire the necessary knowledge to build the expert system. Having a background in chemical engineering greatly assisted the assimilation and organization of the knowledge base.

4.3.3 Knowledge Representation

There are two levels of knowledge represented in this expert system. The first is a plant model based on the chemical/mechanical state model. The second level of knowledge is the relationship between the plant model and the startup procedures.

The plant model was constructed from 11 sets of equipment where one set represented one circuit within the plant unit. Grouping the unit into circuits provides information about the network connection of the equipment. The circuits were based on recommendations from the design and operation teams. These are shown in Table 4.2. Each circuit is made up of

Table 4.2 List of the Circuits in the Plant Unit

Circuit Name	Equipment Within the Circuit
Preheat Circuit	E-01, E-02, E-03, E-04, E-07, E-08, E-09, F-01, Piping
Stripper and Overheads Circuit	D-04, E-10, E-11, E-12, E-18, E-19, T-01, Piping
Stripper Bottoms Circuit	D-06, E-02, E-03, E-04, E-15, E-16, Piping
Debutanizer Circuit	D-05, E-15, E-20, E-21, E-22, T-02, Piping
Vacuum Dryer Circuit	D-07, E-23, T-03, Piping
Vacuum Overheads Circuit	C-05, C-06, D-14, E-24, Piping
Vacuum Bottoms Circuit	E-17, Piping
Recycle Oil Separator Circuit	D-07, Piping
Purge Gas KO Drum Circuit	D-12, Piping
Fuel Gas Circuit	D-08, Piping
Hydrogen Gas Circuit	C-01, C-02, D-03, D-09, D-10, D-11, E-05, E-06, E-07, E-08, E-09, E-10, E-11, F-02, R-01, T-04, Piping

*where: C = compressor, D = drum, E = heat exchanger, F = furnace, R = reactor, T = tower

many pieces of equipment. The circuits are physically separated from each other by block valves. The circuits were modelled by sets of equipment that matched the actual plant equipment. Some heat exchangers are included in two sets. This represents the different circuit flows passing on the shell or tube side of the exchanger. For example, feed oil is passed through the shell side while the reactor outlet is passed through the tube side. The streams transfer heat but do not mix. This heat exchanger is considered part of both the feed oil preheat circuit and reactor effluent circuit.

For this expert system, only major equipment were included in the sets. The more detailed case involving blank locations and minor equipment was not considered in this project. A distinction was made between preparation procedures and startup procedures. Preparation procedures prepare a piece of equipment for the standard startup procedures. The startup procedures are performed to get the unit from its current prepared state to the normal operation state. The level of detail implemented in this model was sufficient for the startup advisory system. If the preparation procedures are to be included then more detailed modelling should be implemented. The preparation procedures include blank placement and removal, equipment opening and closing as well as repair procedures.

The second level of knowledge involved the relationship between the plant model and the startup procedures. These relationships were represented through production rules. There are two types of production rules. The first involves the effects of the operating conditions on the startup procedures. The mode of operation, types of process fluid still in the unit and the type of shutdown are represented in the operating condition rules. The other set of rules cover the effects of opened equipment. A list of opened equipment is built from the user's input. This list is used to determine which circuits are affected. The rules relate circuits with opened equipment to a subset of the startup procedures. These subsets are composed of procedures from several of the startup phases. In total there are 14 production rules covering both sets of rules. Figure 4.2 illustrates the representation format for the

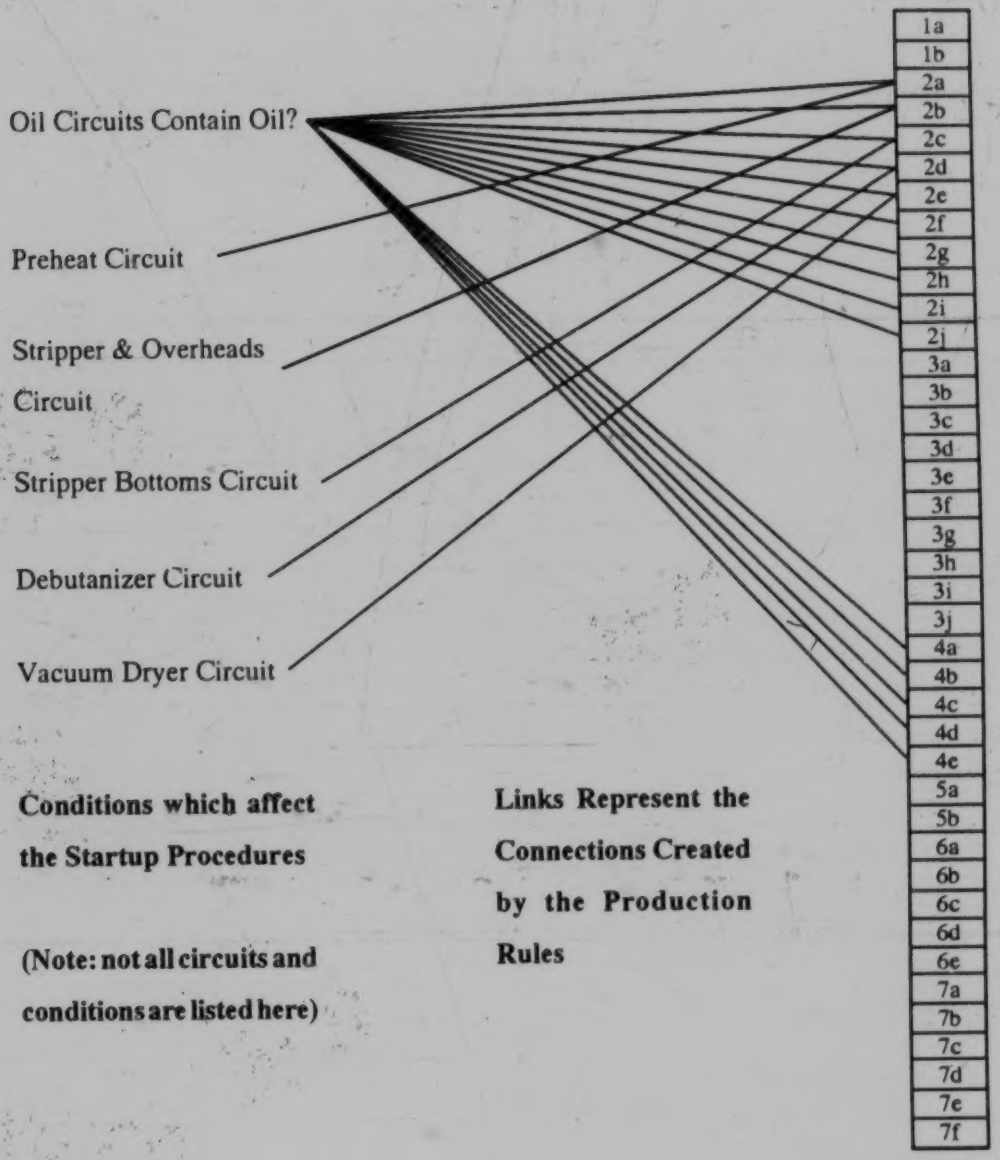


Figure 4.2 Illustration of the Knowledge Structure

knowledge.

The rules governing the effects of opened equipment are implemented using a technique similar to the concept intersection method of Winston (1975). This technique can be viewed as a filtering process. Each rule tests the opened equipment list for equipment within a given circuit. The filtering action is illustrated in Figure 4.3. If information succeeds in passing through the filter then the rule fires and modifies the startup procedures. The filtering process is performed through an intersection operator. The opened equipment list is intersected with a circuit equipment list to determine if equipment within the circuit has been opened. Beyond the implementation details, the rules can be viewed this way:

IF (equipment in circuit X has been opened)
THEN (modify the startup procedures to bring this circuit to normal operation).

The rules are listed in Appendix D. They are presented in pseudo-code format to protect the specific information of the plant.

4.3.4 Hypertext

Hypertext is a technique of linking a keyword or phrase to text that further explains the original keyword or phrase (Gessner, 1990). This technique is mainly used as an information retrieval system. An example of hypertext is an encyclopedia. An encyclopedia contains a large volume of information but is organized into a functional form. The information is grouped into topics according to a short keyword or phrase. At the end of each article is a list of related topics and the keywords to search under. This process of retrieving more information from related sources is fundamental to accessing large amounts of information. Hypertext is a technique to accomplish this task.

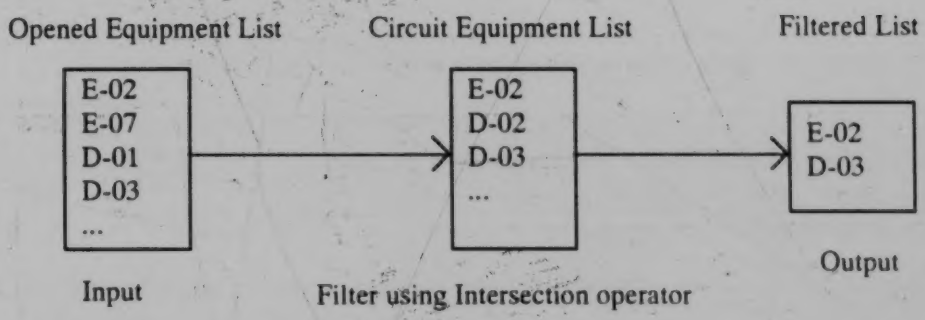


Figure 4.3 Illustration of the Rule Filters

In this expert system, hypertext was used to obtain the startup procedures and their step-by-step instructions. The startup manuals were entered into the computer using a word processor. The text files were then linked through hypertext. Hypertext allowed the operators to select information without knowing the page numbers or searching through unrelated text.

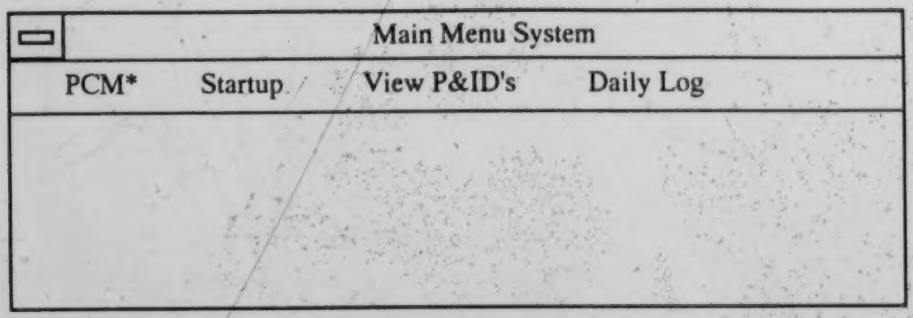
4.3.5 Program Structure

The prototype expert system was developed on a DOS-based microcomputer running Microsoft Windows(TM), (MSW). Its major functions were:

a) to provide decision support during the startup of the plant,
b) to provide a direct connection to the process control computer,
c) to allow viewing of the Process and Instrumentation Diagrams (P&ID's).

This prototype system was built using an expert system shell called KnowledgePro for Windows, which provided immediate access to a graphical user interface. If the expert system was the only function of this prototype then it and the computer would sit idle most of the time. Plant shutdowns and startups happen infrequently and the expert system would have usefulness only during these times. Additional use would come from training of new operators, but this would also be infrequent. To increase the daily usefulness of the system a direct connection to the process control computer was provided. This would allow using the microcomputer as one of the standard terminals used for controlling the plant. Furthermore, providing a means of viewing the P&ID's on screen was found to be a welcome feature.

The main menu system is used to select the major functions. These functions are summarized in Figure 4.4. The process control computer connection was done by using



* PCM = Process Control Mainframe Connection

Figure 4.4 Main Menu System

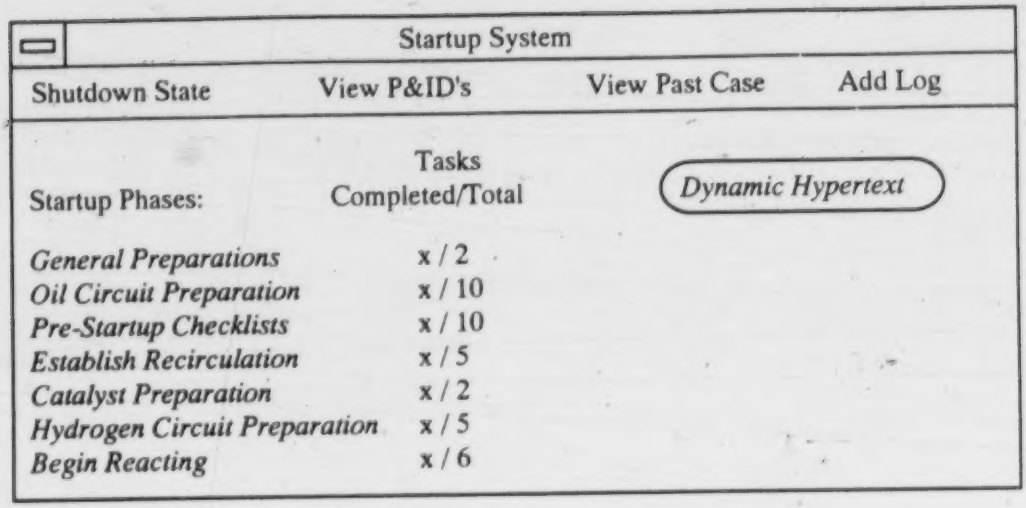
Attachmate's Extra! (TM) for MSW. It acted like a terminal session that could be iconized and brought back just like any other MSW application. To run the session, however, we also needed a hardware card to do the physical connection.

Viewing of the P&ID's was done by shelling out of MSW to Sirlin View's AutoVIEW (TM). AutoVIEW is a non-MSW AutoCAD viewing program. This package provided the ability to display and view AutoCAD drawing files without the overhead needed for editing functions. An important feature of AutoVIEW is the ability to post notes to the drawing. These notes can be viewed quickly and easily by clicking on them with a mouse. This allowed posting of desired modifications to the P&ID's, making notes about special operating conditions as well as equipment safety information.

The Daily Log function was used to either view past log entries or create new ones. The daily logs were saved consecutively to a default file which could be emptied once a month. The Startup function calls up a new window which contains another menu system as well as a hypertext system.

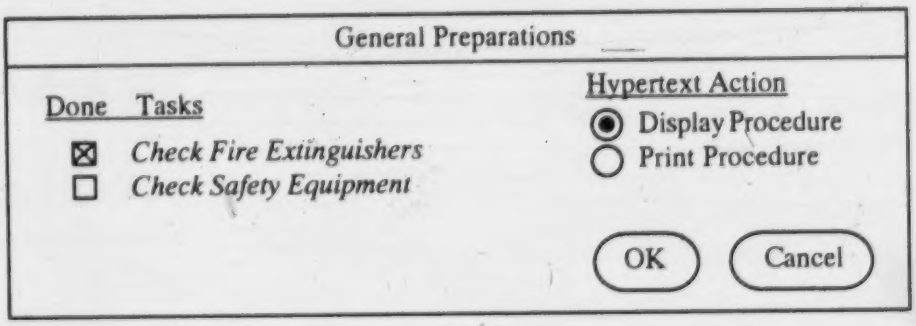
The Startup menu system is shown in Figure 4.5. This menu system provides access to functions that supplement a normal startup sequence. The Shutdown State selection is used to re-enter the shutdown state in case one of the questions was originally answered incorrectly. Viewing of P&ID's is identical to the main menu system. Each startup session is saved in a file. These files contain a summary of the actions taken by the user. It begins with a summary of the shutdown status and follows with a time history of completed startup phases. By using the Add Log function, the user can enter special notes to the startup file. These extra notes could describe difficulties that were encountered and how they were overcome so future startups could have easy access to this information.

The main interaction with the program during a startup is through a hypertext screen



Italics = Hypertext

Figure 4.5 Startup Menu and Hypertext System



Italics = Hypertext

Figure 4.6 Phase and Procedure Hypertext System

that contains a summary of the present state of the startup. There are two areas where hypertext triggers are active (see Figure 4.5). On the left of the screen is a list of the main startup phases. When one of the phase titles is selected by the mouse, a window with the associated procedures for that phase is brought up (see Figure 4.6). Beside each procedure is a check box indicating whether or not the procedure is complete. Furthermore, each of the displayed procedures is also a hypertext trigger. If a procedure is selected, the step-by-step instructions are brought up. The user can choose whether the instructions will go to the screen or printer. The other hypertext section in the startup window is a button. It dynamically changes in function according to the next required step in the startup sequence. By using the dynamic button, an operator can step through the required sequence all using one function. This design was found to be the simplest and most direct approach to the user interface problem.

Figure 4.7 summarizes the overall process of startup operation support. The operator responds to several questions posed by the system. The answers to these questions are used to determine the shutdown state of the unit. Initially, the system assumes that the unit is completely shutdown and all procedures need to be done. The shutdown state and initial startup model are passed into the inferencing section. As the rules in the knowledge base fire, modifications are made to the startup model. When the inferencing is complete, the final startup model is produced. The results are returned to the operator as recommendations on the startup of the unit.

4.4 Evaluation of the Advisory System

This section presents some examples and analysis of the expert systems performance.

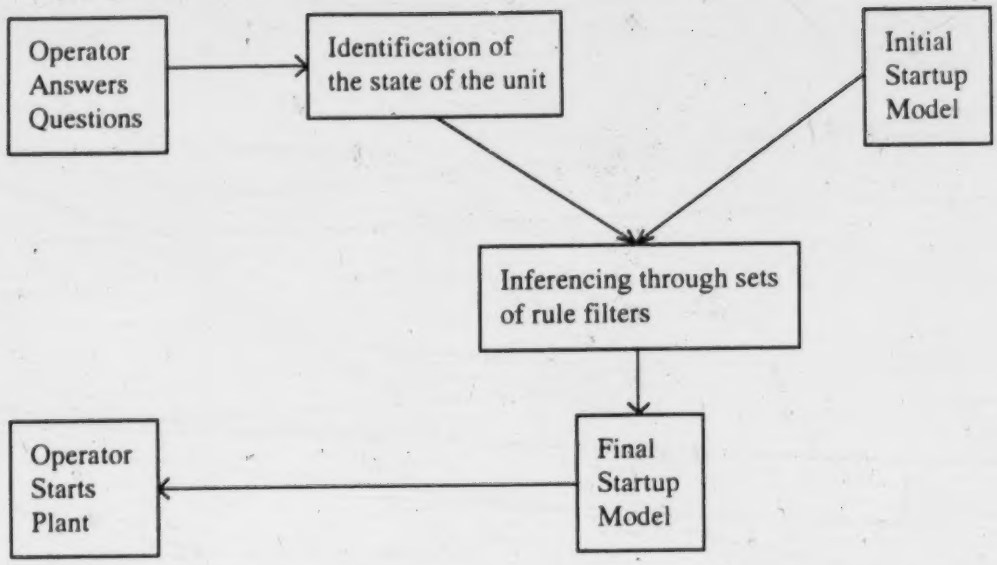

Operator Answers Questions
Identification of the state of the unit
Initial Startup Model
Inferencing through sets of rule filters
Final Startup Model
Operator Starts Plant

Figure 4.7 Knowledge Processing Overview

4.4.1 Examples

The expert system described here has been successfully used in the plant on several occasions. The following example comes from actual testing and use of the system. Some details have been left out to protect proprietary information.

During the summer of 1990, a leak had developed in a line within the Stripper and Overheads circuit. The unit was partially shutdown to repair the leak and to replace a couple of valves on one of the hydrogen compressors at the same time. The input to the expert system was the following:

a) remaining in summer operation
b) oil circuits were not drained
c) hydrogen circuits were not drained
d) compressor C-02 was opened
e) piping in the Stripper and Overheads circuit was opened.

The expert system then made the following recommendation for the startup sequence:

1) General Preparations Phase
   - no action to be done
2) Oil Circuit Preparation Phase
   b) Steam Stripper & Overhead AF&TT procedure
3) Pre-Startup Checklists Phase
   c) Hydrogen Circuits AF&TT procedure
   d) Check Blanks procedure
4) Establish Recirculation Phase
   e) Commission Oil and Vacuum Circuits procedure

5) Catalyst Preparation Phase

- no action to be done

6) Hydrogen Circuit Preparation Phase

e) Establish Hydrogen Recycle procedure

7) Initiate Reacting and Slowly Ramp to Full Capacity Phase

a) Warm both the Oil and Hydrogen Circuits procedure

b) Commission Lean MEA procedure

c) Pump Feed Oil into Unit procedure

d) Swing Feed into Reactor procedure

f) Slowly ramp to Full Capacity procedure

Only 10 of the 40 startup procedures were needed to startup the unit. This is typical for partial shutdowns. The actual startup sequence followed this recommendation for all procedures except one. The deviation between the recommended and actual startup was the Steam Stripper & Overhead AF&TT procedure. The operators only performed a tightness test on the section of pipe where the leak occurred. The contact engineer said that the full circuit test would have been the safer and more thorough approach. The operator in charge, however, was very experienced and had complete confidence in the success of the shortcut taken. The contact engineer then pointed out that another similar shortcut led to a loss in vacuum suction that lasted for weeks. The loss of vacuum was finally traced to an open bleed valve. A full circuit tightness test is used to identify potential problems of this nature. This is a classic case of conflict between experts. Both are correct but one approach may have more advantages than the other. The knowledge base for the developed expert system chose the approach that provided the safest and most thorough sequence. The operators then had the perogative to take shortcuts that they felt were valid.

4.4.2 Evaluation of the Expert System

The startup problem is qualitative in nature. This qualitative nature makes it an excellent problem to solve using artificial intelligence. This same characteristic also makes quantitative analysis an invalid approach for evaluation.

The startup problem has many different solutions. This was illustrated in the last section with the leak repair example. The expert system and the operators were in conflict over the air free and tightness test. Both approaches were correct. However, the expert system's recommendation is more thorough. The shortcut proposed by the operators could lead to further problems like the vacuum leak previously described. In general, the expert system will make the safest recommendation.

One short-coming of this expert system is that primarily one expert's knowledge was incorporated. Several experts' knowledge could be incorporated to provide a more flexible system. The AF&TT shortcut could be implemented safely if a specific set of checks were made prior to startup. However, this project was aimed at identifying the potential use of AI within an industrial setting. The source of expertise for this prototype was primarily from one expert. Concentrating on capturing the knowledge from only one expert helped to accelerate the development of this system. Future development will be aimed at capturing other experts' knowledge.

Another short-coming is the lack of temporal reasoning. This type of reasoning would provide time analysis to the recommendations. Several procedures can be performed in parallel while others can not. This type of knowledge will be added later.

4.5 Discussion

The chemical/mechanical state model provided a means of representing the plant in a compact and useful form. The network connection concept was used to group the equipment into circuits. The mechanical state was used to obtain the chemical state as well as determine the required start procedures. The chemical state provided information about safety hazards and requirements. This model contributed to the successful development of this expert system.

The state of the unit is determined by querying the operator. An alternative would be to derive the shutdown state from process data and the list of work orders on the unit. The process data would provide information on the chemical state and some insight to the mechanical state. The work order list would inform the expert system of which equipment was being opened. The opened equipment list would provide information on the mechanical state. This approach would automate this expert system and bring it another step towards real time operation.

A further issue not pursued in this prototype was the ability to derive the startup procedures from the P&ID's and some knowledge of the chemicals being processed. Several experienced operators said they had performed this derivation for new units. The startup procedures for a unit that is first brought online are usually written by the engineers and operators that will commission the unit. They formulate these procedures based on past experience. Codifying this knowledge would be of benefit during the design stage of a unit. After a few passes of the design of a unit, the startup sequence could be determined to identify areas of operation risk.

4.6 Conclusions

An expert system which provides startup advice has been successfully developed in an industrial setting. Several test runs have been tried and it has performed effectively.

This expert system prototype represents the successful completion of the first step in the development schedule. The success of this phase of development has resulted in additional funding for a multi-man, multi-year extension of this project at the plant. Additionally, an industrial grant to the University of Alberta to support further research.

The retired expert's startup knowledge was captured and automated within an expert system. Expert system technology has proven to be valuable by capturing an expert's knowledge for public use within the company.

Automating written manuals has improved information retrieval efficiency. Hypertext is useful for management and retrieval of large amounts of information. The success of hypertext in this expert system has resulted in a new standard for the company. All written manuals that are entered into computers will be made available to users through hypertext.

The chemical/mechanical state model provided a knowledge representation scheme that was both efficient and understandable. Future development within the company of startup expert systems will incorporate this model.

5. Conclusions

The conclusions presented here are for the general work of the thesis. Conclusions specific to each case study are presented at the end of their respective chapters. Two expert systems were developed in this study. One expert system was built to capture a senior experts' knowledge and retain it for public use. The knowledge captured in this case study was the startup operation for a section of a refinery. Another case study used machine learning to acquire knowledge about the selection of vapor-liquid equilibrium calculation methods. The use of machine learning resulted in strong interdependence of numeric and symbolic information processing.

Problems encountered in chemical engineering are complex and well suited to expert system approaches. The startup expert system demonstrates the success of the expert system objective which is to capture and emulate the expertise of one or more human experts. The plant startup expert's knowledge was effectively represented in the developed expert system. Through several test cases, the expert system demonstrated a high degree of competence. The ability of the system to make correct recommendations for the startup operation has earned it the respect of the plant's operators. The success of the expert system has provided the company with proof of the benefit of expert system technology. Additional funding has been allocated within the company to continue this research and apply its techniques in other areas of the plant.

Another benefit from this startup expert system was a standard for information management. Hypertext was effectively used in this expert system to organize and access the startup manuals. Its success has led the company to adopt hypertext as a plant-wide standard for automation of information processing. This clearly shows that the practical implementation of expert systems can lead to technological benefits beyond.

During the development of the startup expert system a new conceptual model for a chemical process had been developed. It was implemented to structure and design the knowledge base. The model consists of three sections. The first describes the chemical state of the unit. It is desired to move the chemical state from the current state to the desired state. The desired state is the normal operation of the unit. The manipulated variable that is used to drive the chemical state is the mechanical state. The operators make changes to the mechanical state by following procedures to set blanks, valves, etc. Changes in the mechanical state produce changes in the chemical state. The final element of the conceptual model was a description of the relationships between the chemical and mechanical states. This relationship formed the main basis for the knowledge of the expert system.

The vapor-liquid equilibrium expert system built its knowledge from training examples using machine learning. In this manner, it built its own experience from which it could draw during problem-solving sessions. Although expert systems are difficult to evaluate, a test was developed which demonstrated the characteristics of the expert system. The test demonstrated the following points:

1) The expert system gained very little new information after approximately 100 training examples.

2) The accuracies of the expert system for learned and untaught data converged to similar values.

3) The selection of a calculation method to provide the least error in temperature prediction was more successful than for composition.

Point 3 is explained by the relative prediction accuracies shown all the calculation methods. Accuracies for the prediction of composition data were more consistent between the methods. This resulted in increased competition for the selection of a 'best' method. It therefore became more difficult to differentiate between the methods. The accuracies for temperature showed

more variation, allowing a more clear distinction between the methods.
It should be noted that this test does not provide a satisfactory means of testing the worth of the expert system.

The architecture for this expert system can be applied in a large number of property prediction areas. Any physical property which has several methods of prediction could implement this approach.

Expert system development within engineering has shown a definite trend away from traditional approaches. Knowledge engineering was the approach used before the wide-spread availability of expert system shells and techniques. The knowledge engineer is generally ignorant of the domain knowledge he is modelling (Luger and Stubblefield, 1989). This drawback amplifies the problem of building a compact and thorough model of the domain. One solution to this problem was to use automatic knowledge acquisition (Marcus, 1988; Mullarkey, 1990). This approach uses computer programs specifically designed to acquire knowledge from experts and organize it into a programmable form. Another approach is to train domain specialists in the techniques of expert systems and artificial intelligence. Most modern expert systems in engineering are using this approach (Belkin and Stengel, 1990; Gani and O'Connell, 1989; Stephanopoulos and Davis, 1990; Venkatasubramanian, 1985; Venkatasubramanian, 1986). This approach was also used for the two case studies in this thesis. If this trend continues then some formalized training in expert systems for engineers should be established.

References

Abu el Ata-Doss, S. and J. Brunet. (1986). "On-line Expert Supervision for Process Control." Proc. of 25th Conf. on Decision and Control, Athens, Greece, Dec. 1986, pp. 631-632.

Angel, S., A. Marmur and E. Kehat. (1986). "Comparison of Methods of Prediction of Vapor-Liquid Equilibria and Enthalpy in a Distillation Simulation Program." Computers & Chemical Engineering, Vol. 10, No. 2, pp. 169-180.

Banares-Alcantara, R., A. W. Westerberg and M. D. Rychener. (1985). "Development of an Expert System for Physical Property Predictions." Computers & Chemical Engineering, Vol. 9, No. 2, pp. 127-142.

Belkin, B. L. and R. F. Stengel. (1990). "Quantitative Knowledge Acquisition for Expert Systems." Engineering Applications of Artificial Intelligence. Vol. 3, No. 4, pp. 271-281.

Croce, P. A., E. M. Drake, D. E. Wade, R. A. Smith, R. E. Munson. (1988). "Guidelines for Safe Storage and Handling of High Toxic Hazard Materials." Preventing Major Chemical and Related Process Accidents. IChemE Symposium Series No. 110. Institute of Chemical Engineers, Rugby, UK.

Dietterich, T. G. (1986). "Learning at the Knowledge Level". Machine Learning, 1, pp. 287-316.

Gani, R. and J. P. O'Connell. (1989). "A Knowledge Based System for Selection of Thermodynamic Models." Computers & Chemical Engineering, Vol. 13, No. 4/5, pp. 397-404.

Gessner, R., (1990). "Building a Hypertext System." Dr. Dobb's Journal. June, 1990, pp. 22-33.

Gmehling, J., U. Onken and P. Grenzheuser. (1982). Vapor-Liquid Equilibrium Data Collection. Schon & Wetzel GmbH, Frankfurt/Main, F. R. Germany. Published by DECHEMA.

Henley, E. J. and J. D. Seader. (1981). Equilibrium-Stage Separation Operations in Chemical Engineering. John Wiley & Sons, Inc. New York, pp. 282.

Kibler, D. and P. Langley. (1988). "Machine Learning as an Experimental Science." Readings in Machine Learning. Morgan Kaufmann Publishers, Inc. San Mateo, California, pp. 38-43.

Kramer, M. A. (1987). "Malfunction Diagnosis Using Quantitative Models with Non-Boolean Reasoning in Expert Systems." AIChE Journal, Jan. 1987, Vol. 33, No. 1, pp. 130-140.

Kletz, T. A. (1988). What Went Wrong? Case Histories of Process Plant Disasters, 2nd Ed. Gulf Publishing Company, Houston, TX, USA.

Kyle, B. G. (1984). Chemical and Process Thermodynamics. Prentice-Hall, Englewood Cliffs, NJ.

Lazure, N. (1988). Artificial Intelligence Vocabulary. Technology Bulletin 184, Vol. I, Canadian Government Publishing Centre, Ottawa, Canada.

Luger, G. F. and W. A. Stubblefield. (1989). Artificial Intelligence and the Design of Expert

Systems. The Benjamin/Cummings Publishing Company, Inc. Redwood City, California.

Marcus, S. (editor). (1988). Automating Knowledge Acquisition for Expert Systems. Kluwer Academic Publishers, Boston, Mass., USA.

Michalski, R. S. (1983). "A Theory and Methodology of Inductive Learning." Artificial Intelligence, 20, pp. 111-116.

Mullarkey, P. W. (1990). "An Experiment in Direct Knowledge Acquisition." Proceedings, Eighth National Conference on Artificial Intelligence, Volume One, July 29 - Aug 3, 1990. AAAI Press/The MIT Press, Menlo Park, California, USA, pp. 498-504.

Pearson, J. and J. Brazendale. (1988). "Computer Control Chemical Plant - Design and Assessment Framework." Preventing Major Chemical and Related Process Accidents. IChemE Symposium Series No. 110. Institute of Chemical Engineers, Rugby, UK.

Quinlan, J. R. (1986). "Induction of Decision Trees." Machine Learning, 1, pp. 81-106.

Rao, M., T.-S. Jiang and J. J.-P. Tsai. (1989). "Combining Symbolic and Numerical Processing for Real-time Intelligent Control." Eng. Applications of AI, March, 1989, Vol. 2, pp. 19-27.

Reid, R., J. Prausnitz and B. Poling. (1987). The Properties of Gases & Liquids. 4th Ed. McGraw-Hill, New York, NY.

Shavlik, J. and T. G. Dietterich. (1990). Readings in Machine Learning. Morgan Kaufmann Publishers, Inc. San Mateo, California.

Stefik, M. and D. G. Bobrow. (1986). "Object-Oriented Programming: Themes and Variations." The AI Magazine, Dec, pp. 40-62.

Stephanopoulos, G. (1984). Chemical Process Control, An Introduction to Theory and Practice. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, USA.

Stephanopoulos, G. and J. Davis. (1990). Artificial Intelligence in Process Systems Engineering. CAChE Monograph Series, Volume One.

Stroustrup, B. (1986). The C++ Programming Language. Addison-Wesley, Reading, Massachusetts.

Talukdar, S. N., E. Cardozo and L. V. Leao. (1986). "Toast: The Power System Operator's Assistant." IEEE Computer, July 1986, pp. 53-60.

Venkatasubramanian, V. (1985). "An Expert System Based on a Stochastic Parallel Network." Expert Systems in Government Symposium, IEEE Comput. Soc. Press, pp. 77-80.

Venkatasubramanian, V. (1986). "Artificial Intelligence in Process Engineering." Chemical Engineering Education. Vol. XX, No. 4, Fall 1986, pp. 188-192.

Winston, Patrick. (1975). "Learning Structural Descriptions from Examples." The Psychology of Computer Vision. McGraw-Hill, New York, pp. 157-209.

Appendices

This chapter contains the appendices for both case studies. The first three, Appendix A through C, are for the vapor-liquid equilibrium expert system. Appendix D is for the plant startup expert system.

Appendix A: List of the Pure Components Included in the VLE Expert System

1	H2O	Water
2	H3N	Ammonia
3	CCl2O	Phosgene
4	CCl4	Carbon tetrachloride
5	CHCl3	Chloroform
6	CH3Cl	Methyl chloride
7	CH4	Methane
8	CH4O	Methanol
9	C2HCl3	Trichloroethene
10	C2H2	Acetylene
11	C2H3Cl	vinyl chloride
12	C2H3N	Acetonitrile
13	C2H4	Ethylene
14	C2H4Cl2	1,1-dichoroethane
15	C2H4Cl2	1,2-dichloroethane
16	C2H4O	Acetaldehyde
17	C2H4O	Ethylene oxide
18	C2H4O2	Acetic acid
19	C2H4O2	Methyl formate
20	C2H5Cl	Ethyl chloride
21	C2H6	Ethane
22	C2H6O	Dimethyl ether
23	C2H6O	Ethanol
24	C2H7N	Ethyl amine
25	C2H8N2	Ethylenediamine
26	C3H5Cl	Allyl chloride
27	C3H5Cl3	1,2,3-trichloropropane
28	C3H5N	Propionitrile
29	C3H6	Cyclopropane
30	C3H6	Propylene
31	C3H6Cl2	1,2-dichloropropane
32	C3H6O	Acetone

33 C3H6O 1,2-propylene oxide
34 C3H6O2 Propionic acid
35 C3H6O2 Ethyl formate
36 C3H6O2 Methyl acetate
37 C3H7Cl Propyl chloride
38 C3H7Cl Isopropyl chloride
39 C3H8 Propane
40 C3H8O 1-propanol
41 C3H8O Isopropyl alcohol
42 C3H8O Methyl ethyl ether
43 C3H9N N-propyl amine
44 C3H9N Trimethyl amine
45 C4H4O Furan
46 C4H4S Thiophene
47 C4H6 1-butyne
48 C4H6 2-butyne
49 C4H6 1,2-butadiene
50 C4H6 1,3-butadiene
51 C4H6O2 Vinyl acetate
52 C4H8 1-butene
53 C4H8 2-butene,cis
54 C4H8 2-butene,trans
55 C4H8 Isobutylene
56 C4H8O Methyl ethyl ketone
57 C4H8O Tetrahydrofuran
58 C4H8O2 n-butyric acid
59 C4H8O2 1,4-dioxane
60 C4H8O2 Ethyl acetate
61 C4H8O2 Methyl propionate
62 C4H9Cl 1-chlorobutane
63 C4H9Cl 2-chlorobutane
64 C4H9Cl Tert-butyl chloride
65 C4H9N Pyrrolidine
66 C4H9NO Morpholine
67 C4H10 n-butane
68 C4H10 Isobutane
69 C4H10O n-butanol
70 C4H10O 2-butanol
71 C4H10O Isobutanol
72 C4H10O Tert-butanol
73 C4H10O Diethyl ether
74 C4H10O2 1,2-dimethoxyethane

75 C4H11N Diethyl amine
76 C5H5N Pyridine
77 C5H8 1,3-pentadiene,trans
78 C5H8 1,4-pentadiene
79 C5H8 1-pentyne
80 C5H8 2-methyl-1,3-butadiene
81 C5H8O Cyclopentanone
82 C5H10 Cyclopentane
83 C5H10 1-pentene
84 C5H10O Valeraldehyde
85 C5H10O Methyl n-propyl ketone
86 C5H10O Methyl isopropyl ketone
87 C5H10O Diethyl ketone
88 C5H10O2 Isobutyl formate
89 C5H10O2 N-propyl acetate
90 C5H10O2 Ethyl propionate
91 C5H11N Piperidine
92 C5H12 n-pentane
93 C5H12 2-methyl butane
94 C5H12 2,2-dimethlypropane
95 C5H12O 1-pentanol
96 C6H4Cl2 o-dichlorobenzene
97 C6H5Cl Chlorobenzene
98 C6H6 Benzene
99 C6H6O Phenol
100 C6H7N Aniline
101 C6H12 Cyclohexane
102 C6H12 Methylcyclopentane
103 C6H12 1-hexene
104 C6H12O2 n-butyl acetate
105 C6H12O2 Isobutyl acetate
106 C6H12O2 Ethyl butyrate
107 C6H14 n-hexane
108 C6H14O 1-hexanol
109 C6H14O Ethyl butyl ether
110 C6H14O Diisopropyl ether
111 C6H15N Triethylamine
112 C7H6O2 Benzoic acid
113 C7H8 Toluene
114 C7H8O m-cresol
115 C7H14 Methyl cyclohexane
116 C7H14 1-heptene

117	C7H16	n-heptane
118	C7H16	2-methylhexane
119	C7H16	3-methylhexane
120	C7H16	2,2-dimethylpentane
121	C7H16	2,3-dimethylpentane
122	C7H16	2,4-dimethylpentane
123	C7H16	3,3-dimethylpentane
124	C7H16	3-ethylpentane
125	C7H16	2,2,3-trimethylbutane
126	C7H16O	1-heptanol
127	C8H8O	Methyl phenyl ketone
128	C8H8O2	Methyl benzoate
129	C8H10	o-xylene
130	C8H10	m-xylene
131	C8H10	p-xylene
132	C8H10	Ethylbenzene
133	C8H16	Ethylcyclohexane
134	C8H16	1-octene
135	C8H18	n-octane
136	C8H18O	1-octanol
137	C10H8	Naphthalene
138	C10H14	n-butylbenzene
139	C10H22	n-decane
140	C11H10	1-methylnaphthalene
141	C11H10	2-methylnaphthalene

Appendix B: List of the Production Rules for the VLE Expert System

Pure Component Classification Rules

IF compound is one of
7 CH4 Methane
21 C2H6 Ethane
39 C3H8 Propane
67 C4H10 n-butane
68 C4H10 Isobutane
92 C5H12 n-pentane
93 C5H12 2-methyl butane
94 C5H12 2,2-dimethlypropane
107 C6H14 n-hexane
117 C7H16 n-heptane
118 C7H16 2-methylhexane
119 C7H16 3-methylhexane
120 C7H16 2,2-dimethylpentane
121 C7H16 2,3-dimethylpentane
122 C7H16 2,4-dimethylpentane
123 C7H16 3,3-dimethylpentane
124 C7H16 3-ethylpentane
125 C7H16 2,2,3-trimethylbutane
135 C8H18 n-octane
139 C10H22 n-decane
THEN chemical family is
1 Hydrocarbons, Level 2
1 Paraffins, Level 3

IF compound is one of
29 C3H6 Cyclopropane
82 C5H10 Cyclopentane
101 C6H12 Cyclohexane
102 C6H12 Methylcyclopentane
115 C7H14 Methyl cyclohexane
133 C8H16 Ethylcyclohexane
THEN chemical family is
1 Hydrocarbons, Level 2
2 Cycloparaffins, Level 3

IF compound is one of
13 C2H4 Ethylene
30 C3H6 Propylene
49 C4H6 1,2-butadiene
50 C4H6 1,3-butadiene
52 C4H8 1-butene
53 C4H8 2-butene,cis
54 C4H8 2-butene,trans
55 C4H8 Isobutylene
77 C5H8 1,3-pentadiene,trans
78 C5H8 1,4-pentadiene
80 C5H8 2-methyl-1,3-butadiene
83 C5H10 1-pentene
103 C6H12 1-hexene
116 C7H14 1-heptene
134 C8H16 1-octene
THEN chemical family is
1 Hydrocarbons, Level 2
3 Olefins, Level 3

IF compound is one of
10 C2H2 Acetylene
47 C4H6 1-butyne
48 C4H6 2-butyne
79 C5H8 1-pentyne
THEN chemical family is
1 Hydrocarbons, Level 2
4 Acetylenes, Level 3

IF compound is one of
98 C6H6 Benzene
113 C7H8 Toluene
129 C8H10 o-xylene
130 C8H10 m-xylene
131 C8H10 p-xylene
132 C8H10 Ethylbenzene
137 C10H8 Naphthalene
138 C10H14 n-butylbenzene
140 C11H10 1-methylnaphthalene
141 C11H10 2-methylnaphthalene

THEN chemical family is
2 Aromatics, Level 2
5 Aromatics, Level 3

IF compound is one of
96 C6H4Cl2 o-dichlorobenzene
97 C6H5Cl Chlorobenzene
THEN chemical family is
2 Aromatics, Level2
6 Halo-Aromatics, Level 3

IF compound is one of
17 C2H4O Ethylene oxide
33 C3H6O 1,2-propylene oxide
45 C4H4O Furan
46 C4H4S Thiophene
65 C4H9N Pyrrolidine
66 C4H9NO Morpholine
76 C5H5N Pyridine
91 C5H11N Piperidine
THEN chemical family is
2 Aromatics, Level2
7 Hetero-cycles, Level 3

IF compound is one of
4 CCl4 Carbon tetrachloride
5 CHCl3 Chloroform
6 CH3Cl Methyl chloride
9 C2HCl3 Trichloroethene
11 C2H3Cl vinyl chloride
14 C2H4Cl2 1,1-dichoroethane
15 C2H4Cl2 1,2-dichloroethane
20 C2H5Cl Ethyl chloride
27 C3H5Cl3 1,2,3-trichloropropane
31 C3H6Cl2 1,2-dichloropropane
37 C3H7Cl Propyl chloride
38 C3H7Cl Isopropyl chloride
62 C4H9Cl 1-chlorobutane
63 C4H9Cl 2-chlorobutane

64 C4H9Cl Tert-butyl chloride
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
8 Halo-Hydrocarbons, Level 3

IF compound is one of
16 C2H4O Acetaldehyde
32 C3H6O Acetone
56 C4H8O Methyl ethyl ketone
81 C5H8O Cyclopentanone
84 C5H10O Valeraldehyde
85 C5H10O Methyl n-propyl ketone
86 C5H10O Methyl isopropyl ketone
87 C5H10O Diethyl ketone
127 C8H8O Methyl phenyl ketone
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
9 Ketones & Aldehydes, Level 3

IF compound is one of
1 H2O Water
8 CH4O Methanol
23 C2H6O Ethanol
40 C3H8O 1-propanol
41 C3H8O Isopropyl alcohol
69 C4H10O n-butanol
70 C4H10O 2-butanol
71 C4H10O Isobutanol
72 C4H10O Tert-butanol
95 C5H12O 1-pentanol
99 C6H6O Phenol
108 C6H14O 1-hexanol
114 C7H8O m-cresol
126 C7H16O 1-heptanol
136 C8H18O 1-octanol
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
10 Alcohols, Level 3

IF compound is one of
18 C2H4O2 Acetic acid
34 C3H6O2 Propionic acid
58 C4H8O2 n-butyric acid
112 C7H6O2 Benzoic acid
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
11 Acids, Level 3

IF compound is one of
2 H3N Ammonia
24 C2H7N Ethyl amine
25 C2H8N2 Ethylenediamine
43 C3H9N N-propyl amine
44 C3H9N Trimethyl amine
75 C4H11N Diethyl amine
100 C6H7N Aniline
111 C6H15N Triethylamine
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
12 Amines, Level 3

IF compound is one of
3 CCl2O Phosgene
19 C2H4O2 Methyl formate
22 C2H6O Dimethyl ether
35 C3H6O2 Ethyl formate
36 C3H6O2 Methyl acetate
42 C3H8O Methyl ethyl ether
57 C4H8O Tetrahydrofuran
60 C4H8O2 Ethyl acetate
61 C4H8O2 Methyl propionate
73 C4H10O Diethyl ether
74 C4H10O2 1,2-dimethoxyethane
88 C5H10O2 Isobutyl formate
89 C5H10O2 N-propyl acetate
90 C5H10O2 Ethyl propionate
104 C6H12O2 n-butyl acetate
105 C6H12O2 Isobutyl acetate
106 C6H12O2 Ethyl butyrate

109 C6H14O Ethyl butyl ether
110 C6H14O Diisopropyl ether
128 C8H8O2 Methyl benzoate
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
13 Ethers, Esters, Formates, Acetates, etc, Level 3

IF compound is one of
12 C2H3N Acetonitrile
28 C3H5N Propionitrile
THEN chemical family is
3 Substituted Hydrocarbons, Level 2
14 Nitrile, Level 3

VLEC Method Selection Rules

*** Level 1 Rules

If (mixture is any two compounds)
Then (evaluate the best VLEC method using Level 1 Matrices)

*** Level 2 Rules

If (mixture is Hydrocarbon-Hydrocarbon)
Then (evaluate the best VLEC method using Level 2 Matrices for IDs (1, 1))

If (mixture is Hydrocarbon-Aromatic)
Then (evaluate the best VLEC method using Level 2 Matrices for IDs (1, 2))

If (mixture is Hydrocarbon-Substituted Hydrocarbon)
Then (evaluate the best VLEC method using Level 2 Matrices for IDs (1, 3))

If (mixture is Aromatic-Aromatic)
Then (evaluate the best VLEC method using Level 2 Matrices for IDs (2,2))

If (mixture is Aromatic-Substituted Hydrocarbon)
Then (evaluate the best VLEC method using Level 2 Matrices for IDs (2,3))

If (mixture is Substituted Hydrocarbon-Substituted Hydrocabon)

Then (evaluate the best VLEC method using Level 2 Matrices for IDs (3, 3))

*** Level 3 Rules

If (mixture is Paraffin-Paraffin)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 1))

If (mixture is Paraffin-Cycloparaffin)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 2))

If (mixture is Paraffin-Olefin)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 3))

If (mixture is Paraffin-Acetylene)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 4))

If (mixture is Paraffin-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 5))

If (mixture is Paraffin-Halo-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 6))

If (mixture is Paraffin-Hetero-cycle)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 7))

If (mixture is Paraffin-Halo-Hydrocarbon)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 8))

If (mixture is Paraffin-Ketone & Aldehyde)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 9))

If (mixture is Paraffin-Alcohol)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 10))

If (mixture is Paraffin-Acid)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 11))

If (mixture is Paraffin-Amine)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 12))

If (mixture is Paraffin-Ether, Ester, Formate, Acetate, etc.)

Then (evaluate the best VLEC method using Level 3 Matrices for IDs (1, 13))

If (mixture is Paraffin-Nitrile)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 14))

If (mixture is Cycloparaffin-Cycloparaffin)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 2))

If (mixture is Cycloparaffin-Olefin)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 3))

If (mixture is Cycloparaffin-Acetylene)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 4))

If (mixture is Cycloparaffin-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 5))

If (mixture is Cycloparaffin-Halo-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 6))

If (mixture is Cycloparaffin-Hetero-cycle)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 7))

If (mixture is Cycloparaffin-Halo-Hydrocarbon)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 8))

If (mixture is Cycloparaffin-Ketone & Aldehyde)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 9))

If (mixture is Cycloparaffin-Alcohol)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 10))

If (mixture is Cycloparaffin-Acid)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 11))

If (mixture is Cycloparaffin-Amine)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 12))

If (mixture is Cycloparaffin-Ether, Ester, Formate, Acetate, etc.)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 13))

If (mixture is Cycloparaffin-Nitrile)

Then (evaluate the best VLEC method using Level 3 Matrices for IDs (2, 14))

If (mixture is Olefin-Olefin)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 3))

If (mixture is Olefin-Acetylene)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 4))

If (mixture is Olefin-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 5))

If (mixture is Olefin-Halo-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 6))

If (mixture is Olefin-Hetero-cycle)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 7))

If (mixture is Olefin-Halo-Hydrocarbon)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 8))

If (mixture is Olefin-Ketone & Aldehyde)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 9))

If (mixture is Olefin-Alcohol)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 10))

If (mixture is Olefin-Acid)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 11))

If (mixture is Olefin-Amine)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 12))

If (mixture is Olefin-Ether, Ester, Formate, Acetate, etc.)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 13))

If (mixture is Olefin-Nitrile)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (3, 14))

If (mixture is Acetylene-Acetylene)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 4))

If (mixture is Acetylene-Aromatic)

Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 5))

If (mixture is Acetylene-Halo-Aromatic)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 6))

If (mixture is Acetylene-Hetero-cycle)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 7))

If (mixture is Acetylene-Halo-Hydrocarbon)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 8))

If (mixture is Acetylene-Ketone & Aldehyde)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 9))

If (mixture is Acetylene-Alcohol)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 10))

If (mixture is Acetylene-Acid)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 11))

If (mixture is Acetylene-Amine)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 12))

If (mixture is Acetylene-Ether, Ester, Formate, Acetate, etc.)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 13))

If (mixture is Acetylene-Nitrile)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (4, 14))

ETC.

If (mixture is Ether, Ester, Formate, Acetate, etc. - Ether, Ester, Formate, Acetate, etc.)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (13, 13))

If (mixture is Ether, Ester, Formate, Acetate, etc. - Nitrile)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (13, 14))

If (mixture is Nitrile-Nitrile)
Then (evaluate the best VLEC method using Level 3 Matrices for IDs (14, 14))

Appendix C: Source Code for the VLE Expert System

```
// CMP_KB.H

#ifndef CMP_KB_H

class cmp_kb
{
  public:
  unsigned c1_lvl2, c1_lvl3, c2_lvl2, c2_lvl3;
  unsigned level;

  cmp_kb (unsigned ID1, unsigned ID2);
  void reread (unsigned ID1, unsigned ID2);
};

#define CMP_KB_H 1
#endif

// CONSULT.H

#ifndef CONSULT_H

#include "cmp_kb.h"

class consult
{
  cmp_kb *kb1;
  float y_err, yT_err, T_err, Ty_err;
  unsigned num, pem_y, pem_T;

  public:
  consult (char *fname);

  private:
  void pem_kb ();
  unsigned level1_infer ();
  unsigned level2_infer ();
  unsigned level3_infer ();
```

```
  void recommend ();
};

#define CONSULT_H 1
#endif

#ifndef DEFINES_H

#define MAX_PC 141
#define MAX_PEM 15

#define DEFINES_H 1
#endif

// LEARN.H

// This header defines the macine learning object.

#ifndef LEARN_H

#include "cmp_kb.h"
#include "defines.h"

class learn
{
  unsigned ID1;
  unsigned ID2;
  float T;
  float P;
  float x;
  float y;
  float y_errs[MAX_PEM];
  float T_errs[MAX_PEM];
  cmp_kb *kb1;

  public:
  learn (char *filename, int eval);

  private:
  void learn_fxn (char *filename, int eval);
  void learn_level1 ();
```

```
  void learn_level2 ();
  void learn_level3 ();
};

#define LEARN_H 1
#endif

// PC_DB.H

// This header file sets up the pure component database.
#ifndef PC_DB_H

// This structure is the form of the records within the database.
typedef struct
{
  float MW;     // Molecular Weight
  float Tb;     // Boiling Temperature, K
  float Tc;     // Critical Temperature, K
  float Pc;     // Critical Pressure, Pa
  float Vc;     // Critical Molar Volume, m^3/mol
  float wc;     // Acentric Factor
  float Zc;     // Critical Compressibility Factor
  float mu;     // Dipole Moment, debyes
  float cp[4];  // Heat Capacity Polynomial Parameters
  float vp[4];  // Wagner Vapor Pressure Equation Parameters
  float lden;   // Liquid Density, g/ml
  float tden;   // Temperature for Liquid Density, K
  int id;       // component ID
  int family;   // component family
} compdata;

// This object is used to access the database.
class pure_comp_DB
{
  compdata comp1;
  compdata comp2;
  unsigned comp1_ID;
  unsigned comp2_ID;
  char *comp1_nm;
  char *comp2_nm;

  public:
```

```
  pure_comp_DB ();
  pure_comp_DB (unsigned ID);
  pure_comp_DB (unsigned ID1, unsigned ID2);
  ~pure_comp_DB ();

  void get_compdata (unsigned ID, compdata *comp);
  void read_compdata (unsigned ID);
  void print_compdata (unsigned ID);
  void reread (unsigned ID1, unsigned ID2);

  private:

  void read_db_file (unsigned ID);
};

#define PC_DB_H 1
#endif

// PEM_DB.H

// This file defines the Phase Equilibrium Method database.
#ifndef PEM_DB_H

#include <stdio.h>
#include "defines.h"
#include "pc_db.h"

class pem_db
{
  float x;      // liquid molar composition of component 1
  float y;      // calculated vapor molar composition of component 1
  float y_exp;  // vapor molar composition of component 1
  float T;      // temperature of bubble point
  float T_exp;  // temperature of x-y experimental point
  float P;      // pressure of experimental x-y measurement
  float y_errs[MAX_PEM];
  float T_errs[MAX_PEM];
  compdata comp1;  // pure component data
  compdata comp2;  // pure component data
  float wA12, wA21;
  float vA12, vA21;
```

```
  float mA12, mA21;

  public:
  pem_db ();
  ~pem_db ();
  FILE *get_error (pure_comp_DB *comps, float Ti, float Pi, float xi,
    float yi, float *err_in_y, float *err_in_T, FILE *infile);
    // user responsible for freeing memory of return values from get_error
  void print_pem_data (unsigned ID);
  void bubble_pt (unsigned id, void (*get_Ks)(float *, float, float, float,
            float, compdata, compdata));

  private:
  void do_vle_calc ();
  void PengRobinson (unsigned i);
  void SoaveRedlichKwong (unsigned i);
  void RedlichKwong (unsigned i);
  void Wilson (unsigned i);
  void Wil_PengRob (unsigned i);
  void Wil_Soave (unsigned i);
  void Wil_RedKwong (unsigned i);
  void van_Laar (unsigned i);
  void vLr_PengRob (unsigned i);
  void vLr_Soave (unsigned i);
  void vLr_RedKwong (unsigned i);
  void Margules (unsigned i);
  void Mar_PengRob (unsigned i);
  void Mar_Soave (unsigned i);
  void Mar_RedKwong (unsigned i);
};

int cubic_roots (float b, float c, float d, float Z[3]);
float get_Pvp (compdata cmp, float T);

#define PEM_DB_H 1
#endif


// CMP_KB.CPP

#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include "cmp_kb.h"

cmp_kb::cmp_kb (unsigned ID1, unsigned ID2)
{
  reread (ID1, ID2);
}

void cmp_kb::reread (unsigned ID1, unsigned ID2)
{
  FILE *infile;
  char msg[80];
  unsigned i, cmp, c1 = 0, c2 = 0;

  c1_lvl2 = c2_lvl2 = 0;
  infile = fopen ("classify.kb", "rt");

  for (i = 0; i < 14; i++)
  {
    do fgets (msg, 80, infile);
    while (strncmp (msg, "IF", 2) != 0);

    fgets (msg, 80, infile);
    while (strncmp (msg, "THEN", 4) != 0)
    {
      cmp = atoi (msg);
      if (ID1 == cmp) c1 = 1;
      if (ID2 == cmp) c2 = 1;
      fgets (msg, 80, infile);
    }

    if (c1)
    {
      fgets (msg, 80, infile);
      c1_lvl2 = atoi (msg);
      fgets (msg, 80, infile);
      c1_lvl3 = atoi (msg);
    }
    else if (c2)
    {
      fgets (msg, 80, infile);
      c2_lvl2 = atoi (msg);
```

```
      fgets (msg, 80, infile);
      c2_lvl3 = atoi (msg);
    }

    if (c1 && c2)
    {
      c2_lvl2 = c1_lvl2;
      c2_lvl3 = c1_lvl3;
    }

    if (c1_lvl2 && c2_lvl2) break;
    c1 = c2 = 0;
  }

  fclose (infile);

  level = 3;
}

// CONSULT.CPP

#include <stdio.h>
#include <stdlib.h>
#include "consult.h"

consult::consult (char *fname)
{
  FILE *infile;
  char id[10];
  int id1, id2;

  infile = fopen (fname, "rt");
  while (fgets (id, 10, infile))
  {
    id1 = atoi (id);
    fgets (id, 10, infile);
    id2 = atoi (id);
    y_err = yT_err = T_err = Ty_err = 0;
    num = 0;
    printf ("\nMixture IDs: %u, %u\n\n", id1, id2);
    kb1 = new cmp_kb (id1, id2);
```

```
    pem_kb ();
    delete kb1;
  }
}

void consult::pem_kb ()
{
  switch (kb1->level)
  {
    case 3:
      if (level3_infer ()) break;

    case 2:
      if (level2_infer ()) break;

    case 1:
      if (level1_infer ()) break;
  }

  recommend ();
}

unsigned consult::level1_infer ()
{
  char *lv1_file  "all_vle.lv1";
  FILE *infile;
  float y, T;
  unsigned i;

  infile = fopen (lv1_file, "r+b");
  for (i = 0; i < 15; i++)
  {
    fread (&y, sizeof (float), 1, infile);
    fread (&T, sizeof (float), 1, infile);
    fread (&num, sizeof (unsigned), 1, infile);
    if (num == 0) break;
    if (y < y_err || y_err == 0)
    {
      y_err = y;
      yT_err = T;
      pem_y = i;
    }
```

```
    if (T < T_err || T_err == 0)
    {
      T_err = T;
      Ty_err = y;
      pem_T = i;
    }
  }
  fclose (infile);

  return (0);
}

unsigned consult::level2_infer ()
{
  char *lv2_file = "all_vle.lv2";
  unsigned i, offset, low_id, hi_id, row;
  float y, T;
  FILE *infile;

  if (kb1->c1_lvl2 < kb1->c2_lvl2)
  {
    low_id = kb1->c1_lvl2;
    hi_id = kb1->c2_lvl2;
  }
  else
  {
    low_id = kb1->c2_lvl2;
    hi_id = kb1->c1_lvl2;
  }
  for (i = 0, offset = 0, row = 3; i < hi_id-1; i++, row—)
    offset += row;
  offset += low_id-1;
  offset *= 2*sizeof (float) + sizeof (unsigned);
  infile = fopen (lv2_file, "r+b");
  for (i = 0; i < 15; i++, offset += 6*(2*sizeof (float) + sizeof (unsigned)))
  {
    fseek (infile, offset, SEEK_SET);
    fread (&y, sizeof (float), 1, infile);
    fread (&T, sizeof (float), 1, infile);
    fread (&num, sizeof (unsigned), 1, infile);
    if (num == 0) break;
    if (y < y_err || y_err == 0)
```

```
    {
      y_err = y;
      yT_err = T;
      pem_y = i;
    }
    if (T < T_err || T_err == 0)
    {
      T_err = T;
      Ty_err = y;
      pem_T = i;
    }
  }
  fclose (infile);

  return (num);
}

unsigned consult::level3_infer ()
{
  char *lvl3_file[] =
  {
    "pr_eos.lv3",
    "rk_eos.lv3",
    "srk_eos.lv3",
    "wil_act.lv3",
    "wil_pr.lv3",
    "wil_rk.lv3",
    "wil_srk.lv3",
    "vlr_act.lv3",
    "vlr_pr.lv3",
    "vlr_rk.lv3",
    "vlr_srk.lv3",
    "mar_act.lv3",
    "mar_pr.lv3",
    "mar_rk.lv3",
    "mar_srk.lv3"
  };
  unsigned i, offset, low_id, hi_id, row;
  float y, T;
  FILE *infile;

  if (kb1->c1_lvl3 < kb1->c2_lvl3)
```

```
{
  low_id = kb1->c1_lvl3;
  hi_id = kb1->c2_lvl3;
}
else
{
  low_id = kb1->c2_lvl3;
  hi_id = kb1->c1_lvl3;
}
for (i = 0, offset = 0, row = 14; i < hi_id-1; i++, row—)
  offset += row;
offset += low_id-1;
offset *= 2*sizeof (float) + sizeof (unsigned);
for (i = 0; i < 14; i++)
{
  infile = fopen (lvl3_file[i], "r+b");
  fseek (infile, offset, SEEK_SET);
  fread (&y, sizeof (float), 1, infile);
  fread (&T, sizeof (float), 1, infile);
  fread (&num, sizeof (unsigned), 1, infile);
  fclose (infile);
  if (num == 0) break;
  if (y < y_err || y_err == 0)
  {
    y_err = y;
    yT_err = T;
    pem_y = i;
  }
  if (T < T_err || T_err == 0)
  {
    T_err = T;
    Ty_err = y;
    pem_T = i;
  }
 }

 return (num);
}

void consult::recommend ()
{
 char *pem[] =
```

```
{
  "Peng-Robinson",
  "Redlich-Kwong",
  "Soave-Redlich-Kwong",
  "Wilson",
  "Wilson & Peng-Robinson",
  "Wilson & Soave-Redlich-Kwong",
  "Wilson & Redlich-Kwong",
  "van Laar",
  "van Laar & Peng-Robinson",
  "van Laar & Soave-Redlich-Kwong",
  "van Laar & Redlich-Kwong",
  "Margules",
  "Margules & Peng-Robinson",
  "Margules & Soave-Redlich-Kwong",
  "Margules & Redlich-Kwong"
};

if (num == 0)
  printf ("There is no knowledge within the knowledge base.\n");
else
{
  printf ("Recommendation for minimum error in composition:"
       "\n%s\nExpected composition error: •%f mol frac.\n"
       "Expected temperature error: •%f K\n\n", pem[pem_y], y_err,
       yT_err);
  printf ("Recommendation for minimum error in temperture:"
       "\n%s\nExpected composition error: •%f mol frac.\n"
       "Expected temperature error: •%f K\n\n", pem[pem_T], Ty_err,
       T_err);
 }
}


// LEARN.CPP

// This file contains the machine learning routines.

#include <stdio.h>
#include <stdlib.h>
#include "learn.h"
#include "pc_db.h"
```

```
#include "pem_db.h"

void learn::learn_fxn (char *filename, int eval)
{
  FILE *infile;
  pem_db pem;
  pure_comp_DB *pcb;
  char exp_data[80];
  unsigned i, j;

  pcb = new pure_comp_DB (1, 1);
  kb1 = new cmp_kb (1, 1);
  infile = fopen (filename, "rt");

  while (fgets (exp_data, 80, infile))
  {
    ID1 = atoi (exp_data);
    fgets (exp_data, 80, infile);
    ID2 = atoi (exp_data);
    fgets (exp_data, 80, infile);
    T = atof (exp_data);
    fgets (exp_data, 80, infile);
    P = atof (exp_data);
    fgets (exp_data, 80, infile);
    x = atof (exp_data);
    fgets (exp_data, 80, infile);
    y = atof (exp_data);

    pcb->reread (ID1, ID2);
    kb1->reread (ID1, ID2);

    infile = pem.get_error (pcb, T, P, x,.y, y_errs, T_errs, infile);

    switch (eval)
    {
      case 2:
        printf ("\nMixture IDs: %d, %d\n", ID1, ID2);
        for (i = 0, j = 0; i < MAX_PEM; i++)
          if (*(y_errs+i) < *(y_errs+j)) j = i;
        puts ("\nLowest Composition Error:");
        pem.print_pem_data (j+1);
        for (i = 0, j = 0; i < MAX_PEM; i++)
```

```
            if (*(T_errs+i) < *(T_errs+j)) j = i;
          puts ("\nLowest Temperature Error:");
          pem.print_pem_data (j+1);
          break;

        case 1:
          pcb->print_compdata (1);
          pcb->print_compdata (2);
          for (i = 1; i < MAX_PEM+1; i++)
            pem.print_pem_data (i);
          break;

        case 0:
          putchar ('.');
          learn_level1 ();
          learn_level2 ();
          learn_level3 ();
          break;
      }
    }
    delete kb1;
    delete pcb;
    fclose (infile);
}

learn::learn (char *filename, int eval)
{
  learn_fxn (filename, eval);
}

void learn::learn_level1 ()
{
  char *lv1_file = "all_vle.lv1";
  FILE *outfile;
  float y, T;
  long offset = 0;
  unsigned num, i;

  outfile = fopen (lv1_file, "r+b");
  for (i = 0; i < 15; i++, offset += 2*sizeof (float) + sizeof (unsigned))
  {
    fseek (outfile, offset, SEEK_SET);
```

```
      fread (&y, sizeof (float), 1, outfile);
      fread (&T, sizeof (float), 1, outfile);
      fread (&num, sizeof (unsigned), 1, outfile);
      y *= num;
      T *= num;
      num++;
      y = (y+*(y_errs+i))/num;
      T = (T+*(T_errs+i))/num;
      fseek (outfile, offset, SEEK_SET);
      fwrite (&y, sizeof (float), 1, outfile);
      fwrite (&T, sizeof (float), 1, outfile);
      fwrite (&num, sizeof (unsigned), 1, outfile);
    }
    fclose (outfile);
}

void learn::learn_level2 ()
{
    char *lv2_file = "all_vle.lv2";
    unsigned i, offset, low_id, hi_id, row, num;
    float y, T;
    FILE *outfile;

    if (kb1->c1_lvl2 < kb1->c2_lvl2)
    {
      low_id = kb1->c1_lvl2;
      hi_id = kb1->c2_lvl2;
    }
    else
    {
      low_id = kb1->c2_lvl2;
      hi_id = kb1->c1_lvl2;
    }
    for (i = 0, offset = 0, row = 3; i < hi_id-1; i++, row—)
      offset += row;
    offset += low_id-1;
    offset *= 2*sizeof (float) + sizeof (unsigned);
    outfile = fopen (lv2_file, "r+b");
    for (i = 0; i < 15; i++, offset += 6*(2*sizeof (float) + sizeof (unsigned)))
    {
      fseek (outfile, offset, SEEK_SET);
      fread (&y, sizeof (float), 1, outfile);
```

```
      fread (&T, sizeof (float), 1, outfile);
      fread (&num, sizeof (unsigned), 1, outfile);
      y *= num;
      T *= num;
      num++;
      y = (y+*(y_errs+i))/num;
      T = (T+*(T_errs+i))/num;
      fseek (outfile, offset, SEEK_SET);
      fwrite (&y, sizeof (float), 1, outfile);
      fwrite (&T, sizeof (float), 1, outfile);
      fwrite (&num, sizeof (unsigned), 1, outfile);
    }
    fclose (outfile);
}

void learn::learn_level3 ()
{
    char *lvl3_file[] =
    {
      "pr_eos.lv3",
      "rk_eos.lv3",
      "srk_eos.lv3",
      "wil_act.lv3",
      "wil_pr.lv3",
      "wil_rk.lv3",
      "wil_srk.lv3",
      "vlr_act.lv3",
      "vlr_pr.lv3",
      "vlr_rk.lv3",
      "vlr_srk.lv3",
      "mar_act.lv3",
      "mar_pr.lv3",
      "mar_rk.lv3",
      "mar_srk.lv3"
    };
    unsigned i, offset, low_id, hi_id, row, num;
    float y, T;
    FILE *outfile;

    if (kb1->c1_lvl3 < kb1->c2_lvl3)
    {
      low_id = kb1->c1_lvl3;
```

```
    hi_id = kb1->c2_lvl3;
  }
  else
  {
    low_id = kb1->c2_lvl3;
    hi_id = kb1->c1_lvl3;
  }
  for (i = 0, offset = 0, row = 14; i < hi_id-1; i++, row—)
    offset += row;
  offset += low_id-1;
  offset *= 2*sizeof (float) + sizeof (unsigned);
  for (i = 0; i < 14; i++)
  {
    outfile = fopen (lvl3_file[i], "r+b");
    fseek (outfile, offset, SEEK_SET);
    fread (&y, sizeof (float), 1, outfile);
    fread (&T, sizeof (float), 1, outfile);
    fread (&num, sizeof (unsigned), 1, outfile);
    y *= num;
    T *= num;
    num++;
    y = (y+*(y_errs+i))/num;
    T = (T+*(T_errs+i))/num;
    fseek (outfile, offset, SEEK_SET);
    fwrite (&y, sizeof (float), 1, outfile);
    fwrite (&T, sizeof (float), 1, outfile);
    fwrite (&num, sizeof (unsigned), 1, outfile);
    fclose (outfile);
  }
}

// MAR_ACT.CPP

// This file contains the routine for the Margules activity coefficient method.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float mar12, mar21;
```

```
static void get_Ks (float *K, float x, float T, float P, float,
  compdata comp1, compdata comp2)
{
  float act1, act2;

  act1 =  exp (((mar12+3*mar21)*pow (1-x, 2)-4*mar21*pow (1-x, 3))/8.31441/T);
  act2 =  exp (((mar12-3*mar21)*pow (x, 2)+4*mar21*pow (x, 3))/8.31441/T);

  *K =  act1*get_Pvp (comp1, T)/P;
  *(K+1) =  act2*get_Pvp (comp2, T)/P;
}

void pem_db::Margules (unsigned id)
{
  mar12 =  mA12;
  mar21 =  mA21;
  bubble_pt (id, get_Ks);
}

// MAR_PR.CPP

// Margules activity coefficient for liquid phase and Peng-Robinson equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float mar12, mar21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp =  (.37464+1.54226*cmp.wc-.26992*cmp.wc*cmp.wc)*(1.-sqrt (T/cmp.Tc));
  tmp =  31.6087*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
```

```
  return (.64686*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a   get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1.-Bv+2.*Bv, Av-Bv*Bv-2.*Bv-2.*Bv*Bv,
    -Av*Bv+Bv*Bv+Bv*Bv*Bv, Z);
  Zv =  *Z;

  for (int i =  0; i < 2; i++)
  {
    fugv =  exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv/2.828427
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((2.*Zv + 4.828427*Bv)/(2.*Zv - .828427*Bv)));
    *(phiv+i) =  fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2];

  act1 =  exp (((mar12+3*mar21)*pow (1-x, 2)-4*mar21*pow (1-x, 3))/8.31441/T);
  act2 =  exp (((mar12-3*mar21)*pow (x, 2)+4*mar21*pow (x, 3))/8.31441/T);

  get_phiv (phiv, y, T, P, comp1, comp2);
```

```
  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::Mar_PengRob (unsigned id)
{
  mar12 = mA12;
  mar21 = mA21;
  bubble_pt (id, get_Ks);
}

// MAR_RK.CPP

// Margules activity coefficient for liquid phase and Redlich-Kwong equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float mar12, mar21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp = 29.55144*pow (cmp.Tc, 2.5)/cmp.Pc/sqrt (T);
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;
```

```
  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv =  *Z;

  for (int i =  0; i < 2; i++)
  {
    fugv =  exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(phiv+i) =  fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2];

  act1 =  exp (((mar12+3*mar21)*pow (1-x, 2)-4*mar21*pow (1-x, 3))/8.31441/T);
  act2 =  exp (((mar12-3*mar21)*pow (x, 2)+4*mar21*pow (x, 3))/8.31441/T);

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K =  act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) =  act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::Mar_RedKwong (unsigned id)
{
  mar12 =  mA12;
  mar21 =  mA21;
  bubble_pt (id, get_Ks);
}
```

```
// MAR_SRK.CPP

// Margules activity coefficient for liquid phase and Soave equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float mar12, mar21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp =  (.48 + 1.574*cmp.wc - .176*cmp.wc*cmp.wc)*(1. - sqrt (T/cmp.Tc));
  tmp =  29.55144*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
```

```
  Zv = *Z;

  for (int i = 0; i < 2; i++)
  {
    fugv = exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(phiv+i) = fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2];

  act1 = exp (((mar12+3*mar21)*pow (1-x, 2)-4*mar21*pow (1-x, 3))/8.31441/T);
  act2 = exp (((mar12-3*mar21)*pow (x, 2)+4*mar21*pow (x, 3))/8.31441/T);

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::Mar_Soave (unsigned id)
{
  mar12 = mA12;
  mar21 = mA21;
  bubble_pt (id, get_Ks);
}


// MCP.CPP

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "learn.h"
#include "consult.h"

void main (int argc, char *argv[])
```

```
{
  if (argc == 3)
  {
    if (strcmpi (argv[1], "/v") == 0)
    {
      learn verbose (argv[2], 1);
    }
    else if (strcmpi (argv[1], "/e") == 0)
    {
      learn eval (argv[2], 2);
    }
    else if (strcmpi (argv[1], "/c") == 0)
    {
      consult cn (argv[2]);
    }
    else if (strcmpi (argv[1], "/t") == 0)
    {
      learn ml (argv[2], 0);
    }
  }
  else
    puts ("Consult: mcp /c datafile\nEvaluate: mcp /e datafile\n"
        "Verbose Evaluate: mcp /v datafile\nTeach: mcp  /t datafile\n");
}
```

```
#include <stdio.h>
#include <conio.h>
#include "pc_db.h"

void main ()
{
  char name[80], key;
  compdata cmp;
  FILE *nm_file, *pc_file, *nm_out, *pc_out;

  nm_file =  fopen ("pc_db_nm.dat", "rt");
  pc_file =  fopen ("pc_db.dat", "rb");
  nm_out =  fopen ("pc_db_nm.out", "wt");
  pc_out =  fopen ("pc_db.out", "wb");

  for (int i =  0; i < 141; i++)
```

```
  {
    fgets (name, 80, nm_file);
    fread (&cmp, sizeof (compdata), 1, pc_file);
    puts (name);
    puts ("\r\nKeep (y/n)?\r\n\n");
    key = getch ();
    if (key == 'y')
    {
      fwrite (&cmp, sizeof (compdata), 1, pc_out);
      fputs (name, nm_out);
    }
  }
  fclose (pc_out);
  fclose (nm_out);
  fclose (pc_file);
  fclose (nm_file);
}

// PC_DB.CPP

// Pure component database methods.

#include <stdio.h>
#include <mem.h>
#include <stdlib.h>
#include <string.h>
#include "pc_db.h"

// Maximum number of components
#include "defines.h"

pure_comp_DB::pure_comp_DB ()
{
  comp1_ID = comp2_ID = 0;
  comp1_nm = comp2_nm = 0;
}

pure_comp_DB::pure_comp_DB (unsigned ID)
{
  comp1_ID = comp2_ID = 0;
  comp1_nm = comp2_nm = 0;
```

```
  read_db_file (ID);
}

pure_comp_DB::pure_comp_DB (unsigned ID1, unsigned ID2)
{
  comp1_ID = comp2_ID = 0;
  comp1_nm = comp2_nm = 0;
  read_db_file (ID1);
  read_db_file (ID2);
}

pure_comp_DB::~pure_comp_DB ()
{
  comp1_ID = comp2_ID = 0;
  if (comp1_nm) free (comp1_nm);
  if (comp2_nm) free (comp2_nm);
  comp1_nm = comp2_nm = 0;
}

void pure_comp_DB::reread (unsigned ID1, unsigned ID2)
{
  comp1_ID = comp2_ID = 0;
  if (comp1_nm) free (comp1_nm);
  if (comp2_nm) free (comp2_nm);
  comp1_nm = comp2_nm = 0;
  read_db_file (ID1);
  read_db_file (ID2);
}

void pure_comp_DB::read_compdata (unsigned ID)
{
  read_db_file (ID);
}

void pure_comp_DB::get_compdata (unsigned ID, compdata *comp)
{
  if (ID == 1) memcpy (comp, &comp1, sizeof (compdata));
  else if (ID == 2) memcpy (comp, &comp2, sizeof (compdata));
}

void pure_comp_DB::read_db_file (unsigned ID)
{
```

```
unsigned i;
char name[80];

if (ID > MAX_PC) return;
FILE *db_file = fopen ("pc_db.dat", "rb");
FILE *nm_file = fopen ("pc_db_nm.dat", "rt");
if (!comp1_ID)
{
  comp1_ID = ID;
  fseek (db_file, (long) (comp1_ID-1)*sizeof (compdata), SEEK_SET);
  fread (&comp1, sizeof (compdata), 1, db_file);
  for (i = 0; i < ID; i++)
    fgets (name, 80, nm_file);
  comp1_nm = strdup (name);
}
else if (!comp2_ID)
{
  comp2_ID = ID;
  fseek (db_file, (long) (comp2_ID-1)*sizeof (compdata), SEEK_SET);
  fread (&comp2, sizeof (compdata), 1, db_file);
  for (i = 0; i < ID; i++)
    fgets (name, 80, nm_file);
  comp2_nm = strdup (name);
}
fclose (nm_file);
fclose (db_file);
}

void pure_comp_DB::print_compdata (unsigned ID)
{
  if (ID == 1)
    printf ("\nFormula:      Name:\n%sMW : %f\nTb : %f K\nTc : %f K\n"
        "Pc : %f Pa\nVc : %f m^3/mol\nZc : %f\nAc : %f\n",
        comp1_nm, comp1.MW, comp1.Tb, comp1.Tc, comp1.Pc, comp1.Vc,
        comp1.Zc, comp1.wc);
  else if (ID == 2)
    printf ("\nFormula:      Name:\n%sMW : %f\nTb : %f K\nTc : %f K\n"
        "Pc : %f Pa\nVc : %f m^3/mol\nZc : %f\nAc : %f\n",
        comp2_nm, comp2.MW, comp2.Tb, comp2.Tc, comp2.Pc, comp2.Vc,
        comp2.Zc, comp2.wc);
}
```

```
// PEM_DB.CPP

// This file defines the methods for the pem_db object.

#include <mem.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pem_db.h"

static char *vle_nm[] =
{
  "Peng-Robinson",
  "Redlich-Kwong",
  "Soave-Redlich-Kwong",
  "Wilson",
  "Wilson & Peng-Robinson",
  "Wilson & Soave-Redlich-Kwong",
  "Wilson & Redlich-Kwong",
  "van Laar",
  "van Laar & Peng-Robinson",
  "van Laar & Soave-Redlich-Kwong",
  "van Laar & Redlich-Kwong",
  "Margules",
  "Margules & Peng-Robinson",
  "Margules & Soave-Redlich-Kwong",
  "Margules & Redlich-Kwong"
};

pem_db::pem_db ()
{
}

pem_db::~pem_db ()
{
}

void pem_db::print_pem_data (unsigned ID)
{
  printf ("\nPhase Equilibrium Method: %s\nError in y: %f\nError in T: %f\n",
    vle_nm[ID-1], *(y_errs+ID-1), *(T_errs+ID-1));
}
```

```
void pem_db::bubble_pt (unsigned id, void (*get_Ks) (float *, float, float,
  float, float, compdata, compdata))
{
  float K[2], t1, t2, y1, y2, sum_y, y_new;
  unsigned T_iter =  0, y_iter;
  y_new =  y_exp;
  T =  T_exp;

  t1 =  T;
  y_iter =  0;
  do
  {
    y_iter++;
    y =  y_new;
    get_Ks (K, x, T, P, y, comp1, comp2);
    y_new =  *K*x / (*K*x + *(K+1)*(1.-x));
  }
  while (fabs (y - y_new) > .0001 && y_iter < 100);
  y1 =  *K*x + *(K+1)*(1.-x) - 1.;

  t2 =  T += 5;
  y_iter =  0;
  do
  {
    y_iter++;
    y =  y_new;
    get_Ks (K, x, T, P, y, comp1, comp2);
    y_new =  *K*x / (*K*x + *(K+1)*(1.-x));
  }
  while (fabs (y - y_new) > .0001 && y_iter < 100);
  y2 =  *K*x + *(K+1)*(1.-x) - 1.;

  do
  {
    T_iter++;
    T =  t2 - y2 * (t2 - t1) / (y2 - y1);
    y_iter =  0;
    do
    {
      y_iter++;
      y =  y_new;
```

```
      get_Ks (K, x, T, P, y, comp1, comp2);
      y_new =  *K*x / (*K*x + *(K+1)*(1.-x));
    }
    while (fabs (y - y_new) > .0001 && y_iter < 100);
    sum_y =  *K*x + *(K+1)*(1.-x) - 1.;

    if (fabs (y1) < fabs (y2))
    {
      y2 =  sum_y;
      t2 =  T;
    }
    else
    {
      y1 =  sum_y;
      t1 =  T;
    }
  }
  while (fabs (sum_y) > .0001 && fabs (y1-y2) > .0001 && T_iter < 100);

  y =  *K*x / (*K*x + *(K+1)*(1.-x));
  *(y_errs+id) =  fabs (y - y_exp);
  *(T_errs+id) =  fabs (T - T_exp);
}


void pem_db::do_vle_calc ()
{
  unsigned id =  0;
  PengRobinson (id);
  RedlichKwong (++id);
  SoaveRedlichKwong (++id);
  Wilson (++id);
  Wil_PengRob (++id);
  Wil_RedKwong (++id);
  Wil_Soave (++id);
  van_Laar (++id);
  vLr_PengRob (++id);
  vLr_RedKwong (++id);
  vLr_Soave (++id);
  Margules (++id);
  Mar_PengRob (++id);
  Mar_RedKwong (++id);
```

```
  Mar_Soave (++id);
}

FILE *pem_db::get_error (pure_comp_DB *comps, float Ti, float Pi, float xi,
  float yi, float *err_in_y, float *err_in_T, FILE *infile)
{
  char kij[20];
  T = T_exp = Ti;
  P = Pi;
  x = xi;
  y = y_exp = yi;
  comps->get_compdata (1, &comp1);
  comps->get_compdata (2, &comp2);

  fgets (kij, 20, infile);
  wA12 = atof (kij);
  fgets (kij, 20, infile);
  wA21 = atof (kij);

  fgets (kij, 20, infile);
  vA12 = atof (kij);
  fgets (kij, 20, infile);
  vA21 = atof (kij);

  fgets (kij, 20, infile);
  mA12 = atof (kij);
  fgets (kij, 20, infile);
  mA21 = atof (kij);

  do_vle_calc ();
  memcpy (err_in_y, y_errs, MAX_PEM*sizeof (float));
  memcpy (err_in_T, T_errs, MAX_PEM*sizeof (float));

  return (infile);
}


// PR_EOS.CPP

// This file contains the routine for the Peng-Robinson equation of state.

#include <math.h>
```

```
#include "pem_db.h"

int cubic_roots (float b, float c, float d, float Z[3])

/*
    This routine solves a cubic equation of state for the three
    compressibility roots.  An analytical root finding method is
    used (see Perry's, 6th ed, pg 2-15).
    *A =  A*, *B =  B* and u, w are defined in Reid, Prausnitz and Poling
    pg 42-43.  If any root is < 0 or complex a zero is returned instead.
*/

{
  float p, q, cubA, cubB, R, phi;
  int k;

  p =  (3.*c - b*b) / 3.;
  q =  (27.*d - 9.*b*c + 2.*pow(b,3)) / 27.;

  R =  pow(p/3., 3) + q * q / 4.;

  if (R >= 0)
  {
    c =  -q/2. + sqrt (R);
    cubA =  c < 0. ? (-pow(-c, 1./3)) : pow(c, 1./3);
    c =  -q/2. - sqrt (R);
    cubB   c < 0. ? (-pow(-c, 1./3)) : pow(c, 1./3);

    if (R > 0)
    {
      *(Z+2) =  *(Z+1) =  0.;
      *Z =  cubA + cubB - b / 3.;
      return (1);
    }
    else
    {
      *(Z+2) =  cubA + cubB - b / 3.;
      *Z =  -cubA - b / 3.;

      if (*Z > *(Z+2))
        *(Z+1) =  *(Z+2);
      else
```

```
      {
        *(Z+1) = *Z;
        *Z = *(Z+2);
      }

      *(Z+2) = 0.;
      return (2);
    }
  }
  else
  {
    phi = acos (sqrt (-6.75 * q * q / pow(p,3)));

    for (k = 0; k < 3; k++)
      *(Z+k) = (q > 0 ? -2. : 2.) * sqrt (-p/3.) * cos (phi/3. +
        2.094395102 * k) - b / 3.;

    k = (*Z < *(Z+1));
    k = *(Z+k) < *(Z+2) ? 2 : k;
    if (k)
    {
      phi = *Z;
      *Z = *(Z+k);
      *(Z+k) = phi;
    }
    if (*(Z+1) < *(Z+2))
    {
      phi = *(Z+1);
      *(Z+1) = *(Z+2);
      *(Z+2) = phi;
    }

    for (k = 2; k >= 0; k—)
      if (*(Z+k) < 0) *(Z+k) = 0.;
      else break;

    return (k+1);
  }
}

static float get_eos_a (compdata cmp, float T)
{
```

```
  float tmp;

  tmp = (.37464+1.54226*cmp.wc-.26992*cmp.wc*cmp.wc)*(1.-sqrt (T/cmp.Tc));
  tmp = 31.6087*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.64686*cmp.Tc/cmp.Pc);
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float Z[3], Al, Av, Bl, Bv, fugv, fugl, Zl, Zv;
  float eos_a[2], eos_b[2], ax, ay, bx, by;
  int k;

  *eos_a = get_eos_a (comp1, T);
  *(eos_a+1) = get_eos_a (comp2, T);
  *eos_b = get_eos_b (comp1);
  *(eos_b+1) = get_eos_b (comp2);

  ax = pow (x*sqrt (*eos_a) + (1.-x)*sqrt (*(eos_a+1)), 2);
  bx = x**eos_b + (1.-x)**(eos_b+1);
  ay = pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by = y**eos_b + (1.-y)**(eos_b+1);

  Al = .014465622 * ax * P / T / T;
  Bl = .120273116 * bx * P / T;
  k = cubic_roots (-1.-Bl+2.*Bl, Al-Bl*Bl-2.*Bl-2.*Bl*Bl,
    -Al*Bl+Bl*Bl+Bl*Bl*Bl, Z);
  Zl = *(Z+k-1);

  Av = Al*ay/ax;
  Bv = Bl*by/bx;
  k = cubic_roots (-1.-Bv+2.*Bv, Av-Bv*Bv-2.*Bv-2.*Bv*Bv,
    -Av*Bv+Bv*Bv+Bv*Bv*Bv, Z);
  Zv = *Z;

  for (int i = 0; i < 2; i++)
```

```
  {
    fugl =  exp (*(eos_b+i)/bx *(Zl - 1.) - log (Zl-Bl) + Al/Bl/2.828427
      * (*(eos_b+i)/bx - 2.*sqrt (*(eos_a+i)/ax)) *
      log ((2.*Zl + 4.828427*Bl)/(2.*Zl - .828427*Bl)));
    *(K+i) =  fugl;

    fugv =  exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv/2.828427
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((2.*Zv + 4.828427*Bv)/(2.*Zv - .828427*Bv)));
    *(K+i) /=  fugv;
  }
}

void pem_db::PengRobinson (unsigned id)
{
  bubble_pt (id, get_Ks);
}


// RK_EOS.CPP

// This file contains the routine for the Redlich-Kwong equation of state.

#include <math.h>
#include "pem_db.h"

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp =  29.55144*pow (cmp.Tc, 2.5)/cmp.Pc/sqrt (T);
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
```

```
  float Z[3], Al, Av, Bl, Bv, fugv, fugl, Zl, Zv;
  float eos_a[2], eos_b[2], ax, ay, bx, by;
  int k;

  *eos_a = get_eos_a (comp1, T);
  *(eos_a+1) = get_eos_a (comp2, T);
  *eos_b = get_eos_b (comp1);
  *(eos_b+1) = get_eos_b (comp2);

  ax = pow (x*sqrt (*eos_a) + (1.-x)*sqrt (*(eos_a+1)), 2);
  bx = x**eos_b + (1.-x)**(eos_b+1);
  ay = pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by = y**eos_b + (1.-y)**(eos_b+1);

  Al = .014465622 * ax * P / T / T;
  Bl = .120273116 * bx * P / T;
  k = cubic_roots (-1., Al-Bl-Bl*Bl, -Al*Bl, Z);
  Zl = *(Z+k-1);

  Av = Al*ay/ax;
  Bv = Bl*by/bx;
  k = cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv = *Z;

  for (int i = 0; i < 2; i++)
  {
    fugl = exp (*(eos_b+i)/bx *(Zl - 1.) - log (Zl-Bl) + Al/Bl
      * (*(eos_b+i)/bx - 2.*sqrt (*(eos_a+i)/ax)) *
      log ((Zl + Bl)/Zl));
    *(K+i) = fugl;

    fugv = exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(K+i) /= fugv;
  }
}

void pem_db::RedlichKwong (unsigned id)
{
  bubble_pt (id, get_Ks);
}
```

```
// SRK_EOS.CPP

// This file contains the routine for the Soave-Redlich-Kwong equation of state

#include <math.h>
#include "pem_db.h"

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp =  (.48 + 1.574*cmp.wc - .176*cmp.wc*cmp.wc)*(1. - sqrt (T/cmp.Tc));
  tmp =  29.55144*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float Z[3], Al, Av, Bl, Bv, fugv, fugl, Zl, Zv;
  float eos_a[2], eos_b[2], ax, ay, bx, by;
  int k;

  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ax =  pow (x*sqrt (*eos_a) + (1.-x)*sqrt (*(eos_a+1)), 2);
  bx =  x**eos_b + (1.-x)**(eos_b+1);
  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Al =  .014465622 * ax * P / T / T;
  Bl =  .120273116 * bx * P / T;
  k =  cubic_roots (-1., Al-Bl-Bl*Bl, -Al*Bl, Z);
  Zl =  *(Z+k-1);
```

```
  Av = Al*ay/ax;
  Bv = Bl*by/bx;
  k = cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv = *Z;

  for (int i = 0; i < 2; i++)
  {
    fugl = exp (*(eos_b+i)/bx *(Zl - 1.) - log (Zl-Bl) + Al/Bl
      * (*(eos_b+i)/bx - 2.*sqrt (*(eos_a+i)/ax)) *
      log ((Zl + Bl)/Zl));
    *(K+i) = fugl;

    fugv = exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(K+i) /= fugv;
  }
}

void pem_db::SoaveRedlichKwong (unsigned id)
{
  bubble_pt (id, get_Ks);
}

// VLR_ACT.CPP

// This file contains the routine for the van Laar activity coefficient method.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float vlr12, vlr21;

static void get_Ks (float *K, float x, float T, float P, float,
  compdata comp1, compdata comp2)
{
  float act1, act2;

  act1 = exp (vlr12/8.31441/T/pow (1+vlr12*x/vlr21/(1-x), 2));
```

```
  act2 = exp (vlr21/8.31441/T/pow (1+vlr21*(1-x)/vlr12/x, 2));

  *K = act1*get_Pvp (comp1, T)/P;
  *(K+1) = act2*get_Pvp (comp2, T)/P;
}

void pem_db::van_Laar (unsigned id)
{
  vlr12 = vA12;
  vlr21 = vA21;
  bubble_pt (id, get_Ks);
}


// VLR_PR.CPP

// Van Laar activity coefficient for liquid phase and Peng-Robinson equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float vlr12, vlr21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp = (.37464+1.54226*cmp.wc-.26992*cmp.wc*cmp.wc)*(1.-sqrt (T/cmp.Tc));
  tmp = 31.6087*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.64686*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
```

```
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a = get_eos_a (comp1, T);
  *(eos_a+1) = get_eos_a (comp2, T);
  *eos_b = get_eos_b (comp1);
  *(eos_b+1) = get_eos_b (comp2);

  ay = pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by = y**eos_b + (1.-y)**(eos_b+1);

  Av = .014465622 * ay * P / T / T;
  Bv = .120273116 * by * P / T;
  (void) cubic_roots (-1.-Bv+2.*Bv, Av-Bv*Bv-2.*Bv-2.*Bv*Bv,
    -Av*Bv+Bv*Bv+Bv*Bv*Bv, Z);
  Zv = *Z;

  for (int i = 0; i < 2; i++)
  {
    fugv = exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv/2.828427
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((2.*Zv + 4.828427*Bv)/(2.*Zv - .828427*Bv)));
    *(phiv+i) = fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2];

  act1 = exp (vlr12/8.31441/T/pow (1+vlr12*x/vlr21/(1-x), 2));
  act2 = exp (vlr21/8.31441/T/pow (1+vlr21*(1-x)/vlr12/x, 2));

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::vLr_PengRob (unsigned id)
{
```

```
  vlr12 = vA12;
  vlr21 = vA21;
  bubble_pt (id, get_Ks);
}

// VLR_RK.CPP

// Van Laar activity coefficient for liquid phase and Redlich-Kwong equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float vlr12, vlr21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp = 29.55144*pow (cmp.Tc, 2.5)/cmp.Pc/sqrt (T);
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a = get_eos_a (comp1, T);
  *(eos_a+1) = get_eos_a (comp2, T);
  *eos_b = get_eos_b (comp1);
  *(eos_b+1) = get_eos_b (comp2);

  ay = pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
```

```
  by = y**eos_b + (1.-y)**(eos_b+1);

  Av = .014465622 * ay * P / T / T;
  Bv = .120273116 * by * P / T;
  (void) cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv = *Z;

  for (int i = 0; i < 2; i++)
  {
    fugv = exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(phiv+i) = fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2];

  act1 = exp (vlr12/8.31441/T/pow (1+vlr12*x/vlr21/(1-x), 2));
  act2 = exp (vlr21/8.31441/T/pow (1+vlr21*(1-x)/vlr12/x, 2));

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::vLr_RedKwong (unsigned id)
{
  vlr12 = vA12;
  vlr21 = vA21;
  bubble_pt (id, get_Ks);
}


// VLR_PR.CPP

// Van Laar activity coefficient for liquid phase and Soave equation of
// state for vapor phase.
```

```
#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float vlr12, vlr21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp =  (.48 + 1.574*cmp.wc - .176*cmp.wc*cmp.wc)*(1. - sqrt (T/cmp.Tc));
  tmp =  29.55144*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv   *Z;

  for (int i =  0; i < 2; i++)
  {
```

```
    fugv = exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(phiv+i) = fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2];

  act1 = exp (vlr12/8.31441/T/pow (1+vlr12*x/vlr21/(1-x), 2));
  act2 = exp (vlr21/8.31441/T/pow (1+vlr21*(1-x)/vlr12/x, 2));

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::vLr_Soave (unsigned id)
{
  vlr12 = vA12;
  vlr21 = vA21;
  bubble_pt (id, get_Ks);
}


// WIL_ACT.CPP

// This file contains the routine for the Wilson activity coefficient method.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float wil12, wil21;

float get_Pvp (compdata cmp, float T)
{
  float a = 1 - T / cmp.Tc, Pvp;
```

```
  Pvp =  cmp.vp[0]*a + cmp.vp[1]*pow (a, 1.5) + cmp.vp[2]*pow (a, 3)
    + cmp.vp[3]*pow (a, 6);
  Pvp /=  1 - a;
  Pvp =  exp (Pvp)*cmp.Pc;

  return (Pvp);
}
static void get_Ks (float *K, float x, float T, float P, float,
  compdata comp1, compdata comp2)
{
  float act1, act2, A12, A21;

  A12 =  comp2.lden/comp1.lden*exp (-wil12/8.31441/T);
  A21 =  comp1.lden/comp2.lden*exp (-wil21/8.31441/T);

  act1 =  exp (-log (x+A12*(1-x))+(1-x)*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));
  act2 =  exp (-log (1-x+A21*x)-x*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));

  *K =  act1*get_Pvp (comp1, T)/P;
  *(K+1) =  act2*get_Pvp (comp2, T)/P;
}

void pem_db::Wilson (unsigned id)
{
  wil12 =  wA12;
  wil21 =  wA21;
  bubble_pt (id, get_Ks);
}


// WIL_PR.CPP

// Wilson activity coefficient for liquid phase and Peng-Robinson equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float wil12, wil21;
```

```
static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp =  (.37464+1.54226*cmp.wc-.26992*cmp.wc*cmp.wc)*(1.-sqrt (T/cmp.Tc));
  tmp =  31.6087*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.64686*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1.-Bv+2.*Bv, Av-Bv*Bv-2.*Bv-2.*Bv*Bv,
    -Av*Bv+Bv*Bv+Bv*Bv*Bv, Z);
  Zv =  *Z;

  for (int i =  0; i < 2; i++)
  {
    fugv =  exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv/2.828427
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((2.*Zv + 4.828427*Bv)/(2.*Zv - .828427*Bv)));
    *(phiv+i) =  fugv;
  }
}
```

```
static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2], A12, A21;

  A12 = comp2.lden/comp1.lden*exp (-wil12/8.31441/T);
  A21 = comp1.lden/comp2.lden*exp (-wil21/8.31441/T);

  act1 = exp (-log (x+A12*(1-x))+(1-x)*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));
  act2 = exp (-log (1-x+A21*x)-x*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::Wil_PengRob (unsigned id)
{
  wil12 = wA12;
  wil21 = wA21;
  bubble_pt (id, get_Ks);
}


// WIL_PR.CPP

// Wilson activity coefficient for liquid phase and Peng-Robinson equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float wil12, wil21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp = 29.55144*pow (cmp.Tc, 2.5)/cmp.Pc/sqrt (T);
  return (tmp);
```

```
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv =  *Z;

  for (int i =  0; i < 2; i++)
  {
    fugv =  exp (*(eos_b+i)/by *(Zv - 1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(phiv+i) =  fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2], A12, A21;

  A12 =  comp2.lden/comp1.lden*exp (-wil12/8.31441/T);
  A21 =  comp1.lden/comp2.lden*exp (-wil21/8.31441/T);
```

```
  act1 = exp (-log (x+A12*(1-x))+(1-x)*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));
  act2 = exp (-log (1-x+A21*x)-x*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));

  get_phiv (phiv, y, T, P, comp1, comp2);

  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::Wil_RedKwong (unsigned id)
{
  wil12 = wA12;
  wil21 = wA21;
  bubble_pt (id, get_Ks);
}


// WIL_PR.CPP

// Wilson activity coefficient for liquid phase and Peng-Robinson equation of
//  state for vapor phase.

#include <math.h>
#include <stdlib.h>
#include "pem_db.h"

static float wil12, wil21;

static float get_eos_a (compdata cmp, float T)
{
  float tmp;

  tmp = (.48 + 1.574*cmp.wc - .176*cmp.wc*cmp.wc)*(1. - sqrt (T/cmp.Tc));
  tmp = 29.55144*cmp.Tc*cmp.Tc*pow ((1. + tmp), 2)/cmp.Pc;
  return (tmp);
}

static float get_eos_b (compdata cmp)
{
  return (.72036*cmp.Tc/cmp.Pc);
```

```
}

static void get_phiv (float *phiv, float y, float T, float P,
  compdata comp1, compdata comp2)
{
  float Z[3], Av, Bv, fugv, Zv;
  float eos_a[2], eos_b[2], ay, by;

  *eos_a =  get_eos_a (comp1, T);
  *(eos_a+1) =  get_eos_a (comp2, T);
  *eos_b =  get_eos_b (comp1);
  *(eos_b+1) =  get_eos_b (comp2);

  ay =  pow (y*sqrt (*eos_a) + (1.-y)*sqrt (*(eos_a+1)), 2);
  by =  y**eos_b + (1.-y)**(eos_b+1);

  Av =  .014465622 * ay * P / T / T;
  Bv =  .120273116 * by * P / T;
  (void) cubic_roots (-1., Av-Bv-Bv*Bv, -Av*Bv, Z);
  Zv =  *Z;

  for (int i =  0; i < 2; i++)
  {
    fugv =  exp (*(eos_b+i)/by *(Zv -1.) - log (Zv-Bv) + Av/Bv
      * (*(eos_b+i)/by - 2.*sqrt (*(eos_a+i)/ay)) *
      log ((Zv + Bv)/Zv));
    *(phiv+i) =  fugv;
  }
}

static void get_Ks (float *K, float x, float T, float P, float y,
  compdata comp1, compdata comp2)
{
  float act1, act2, phiv[2], A12, A21;

  A12 =  comp2.lden/comp1.lden*exp (-wil12/8.31441/T);
  A21 =  comp1.lden/comp2.lden*exp (-wil21/8.31441/T);

  act1 =  exp (-log (x+A12*(1-x))+(1-x)*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));
  act2 =  exp (-log (1-x+A21*x)-x*(A12/(x+A21*(1-x))-A21/(x*A21+1-x)));

  get_phiv (phiv, y, T, P, comp1, comp2);
```

```
  *K = act1*get_Pvp (comp1, T)/P/ *phiv;
  *(K+1) = act2*get_Pvp (comp2, T)/P/ *(phiv+1);
}

void pem_db::Wil_Soave (unsigned id)
{
  wil12 = wA12;
  wil21 = wA21;
  bubble_pt (id, get_Ks);
}
```

Appendix D: Production Rules for the Plant Startup Expert System

```
Rule 01
If
  (unit is in same operation (Summer or Winter) both before and after shutdown)
Then
  (set
    General Preparations Phase (1) to Done
    Pre-Startup Checklist Phase (3) to Done
    Catalyst Preparation Phase (5) to Done
  )
Else
  If
    (going to Winter operation)
  Then
    (set
      Caustic Treating Procedure (5a) to Done
    )
  Else
    (set
      Debutanizer Procedure (2d)
      Back in Naphtha into Debutanizer Procedure (4a) to Done
      Presulphiding Procedure (5b) to Done
      Establish Debutanizer Reflux Procedure (7e) to Done
    )


Rule 02
If
  (oil circuits are still under oil)
Then
  (set
    Establish Recirculation Phase (4) to Done
    Oil Circuit Preparation Phase (2) to Done
  )


Rule 03
If
  (hydrogen circuits are still under hydrogen)
Then
```

(set
Hydrogen Circuits AF&TT Procedure (3c) to Done
Hydrogen Circuit Preparation Phase (6) to Done
)

Rule 04
If
(no opened equipment on the Pre-Heat Circuit)
Then
(set
Oil Feed Circuit Procedure (2a) to Done
)
Else
(set
Oil Feed Circuit Procedure (2a) to Not Done
)

Rule 05
If
(no opened equipment on the Stripper & Overheads Circuit)
Then
(set
Steam Stripper & Overheads Procedure (2b) to Done
)
Else
(set
Steam Stripper & Overheads Procedure (2b) to Not Done
)

Rule 06
If
(no opened equipment on the Stripper Bottoms Circuit)
Then
(set
Stripper Bottoms Procedure (2c) to Done
)
Else
(set
Stripper Bottoms Procedure (2c) to Not Done

```
  )

Rule 07
If
  (going to Winter operation)
Then
  If
    (no opened equipment on the Debutanizer Circuit)
  Then
    (set
      Debutanizer Procedure (2d) to Done
    )
  Else
    (set
      Debutanizer Procedure (2d) to Not Done
    )

Rule 08
If
  (no opened equipment on the Vacuum Dryer Circuit)
Then
  (set
    Vacuum Dryer Procedure (2e) to Done
  )
Else
  (set
    Vacuum Dryer Procedure (2e) to Not Done
  )

Rule 09
If
  (no opened equipment on the Vacuum Overheads Circuit)
Then
  (set
    Vacuum Overheads Procedure (2f) to Done
  )
Else
  (set
    Vacuum Overheads Procedure (2f) to Not Done
```

)

Rule 10
If
(no opened equipment on the Vacuum Bottoms Circuit)
Then
(set
Vacuum Bottoms Procedure (2g) to Done
)
Else
(set
Vacuum Bottoms Procedure (2g) to Not Done
)

Rule 11
If
(no opened equipment on the Recycle Oil Separator Circuit)
Then
(set
Recycle Oil Separator Procedure (2h) to Done
)
Else
(set
Recycle Oil Separator Procedure (2h) to Not Done
)

Rule 12
If
(no opened equipment on the Purge Gas KO Drum Circuit)
Then
(set
Purge Gas KO Drum Procedure (2i) to Done
)
Else
(set
Purge Gas KO Drum Procedure (2i) to Not Done
)

Rule 13
If
  (no opened equipment on the Fuel Gas Circuit)
Then
  (set
    Fuel Gas Circuit Procedure (2j) to Done
)
Else
  (set
    Fuel Gas Circuit Procedure (2j) to Not Done
)

Rule 14
If
  (no work is done on either furnace's refractory)
Then
  (set
    Furnace Dryout Procedure (3b) to Done
)
Else
  (set
    Furnace Dryout Procedure (3b) to Not Done
)